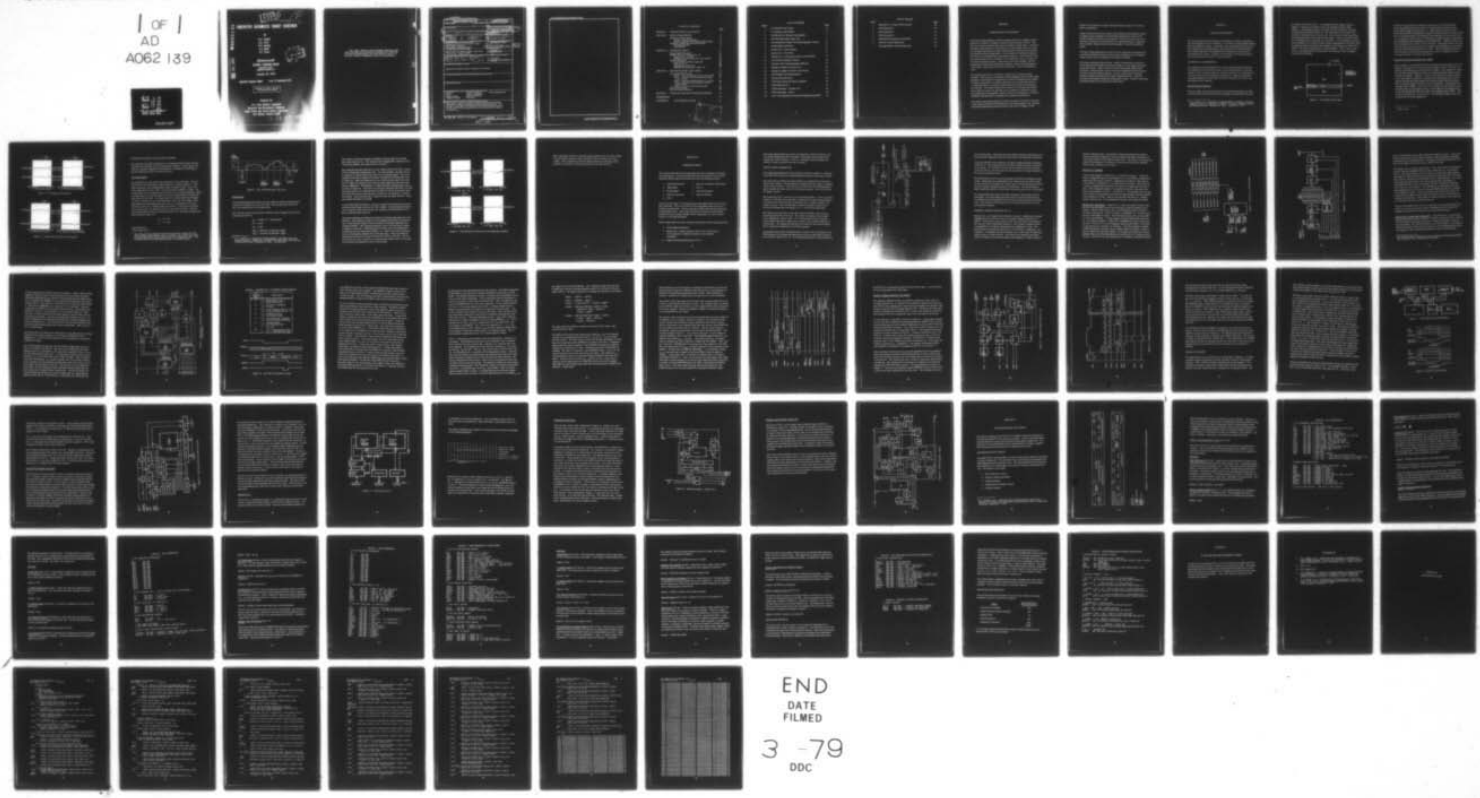
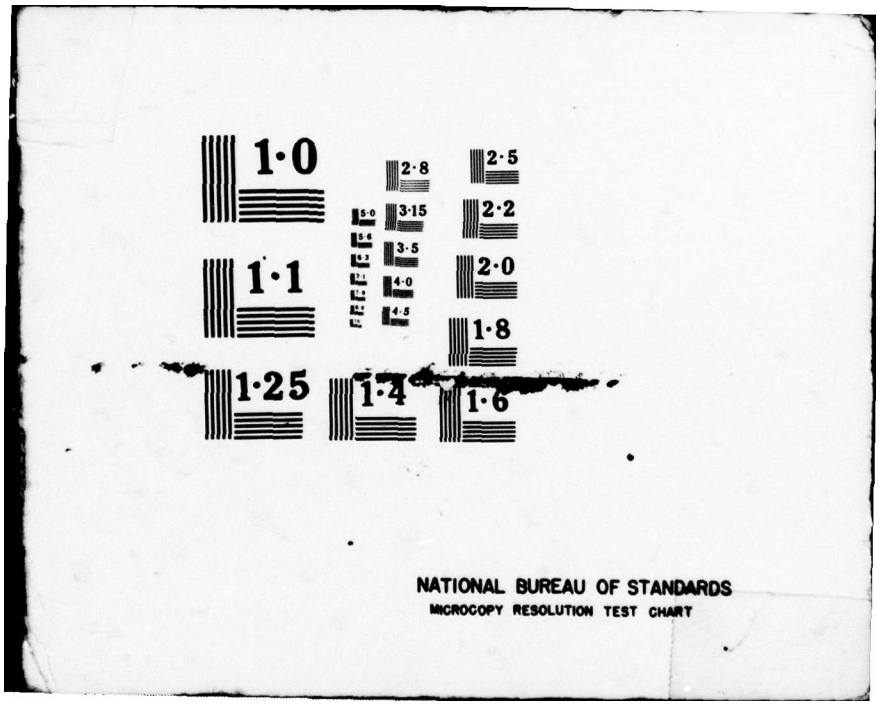
AD-A062 139 HONEYWELL INC MINNEAPOLIS MINN SYSTEMS AND RESEARCH --ETC F/G 17/5
PROTOTYPE AUTOMATIC TARGET SCREENER.(U)
OCT 78 D E SOLAND, G J PROM, R C FITCH DAAK70-77-C-0248

UNCLASSIFIED 78SRC54 NL

1 OF 1
AD
A062 139



END
DATE
FILMED
3 -79
DDC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
5·0
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

PROTOTYPE AUTOMATIC TARGET SCREENER

By

D.E. Soland
R.C. Fitch
D.V. Serreyn
T.G. Kopet
G.J. Prom

Honeywell

SYSTEMS & RESEARCH CENTER
2600 RIDGWAY PARKWAY
MINNEAPOLIS, MINNESOTA 55413

October 20, 1978

Quarterly Progress Report 1 July—30 September 1978

Prepared for

U.S. Army Mobility Equipment
Research and Development Command,
Night Vision and Electro-Optics Laboratory
Fort Belvoir, Virginia 22060

"The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official department of the Army position, policy, or decision, unless so designated by other documentations."

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(6) PROTOTYPE AUTOMATIC TARGET SCREENER

(9) 5. TYPE OF REPORT & PERIOD COVERED
Quarterly Progress Report no. 4,
1 July - 30 September, 1978

6. PERFORMING ORG. REPORT NUMBER
(14) 78SRC54

7. AUTHOR(s)
(10) D. E. Soland, D. V. Serreyn
G. J. Prom, T. G. Kopet
R. C. Fitch

8. CONTRACT OR GRANT NUMBER(s)
(15) DAAK70-77-C-0248

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Honeywell Systems and Research Center
2600 Ridgway Parkway
Minneapolis, Minnesota 55413

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
(16) 62637 10DK70 14 010CJ
(17) 14

11. CONTROLLING OFFICE NAME AND ADDRESS
Night Vision and Electro-Optics Laboratory
Fort Belvoir, Virginia 22060

12. REPORT DATE
October 16, 1978

13. NUMBER OF PAGES
72

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
(11) 20 Oct 78
(12) 74 p.

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Infrared	Target recognition	Image enhancement
FLIR	Pattern recognition	
Target cueing	Image processing	
Target screening	Real time	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is the fourth quarterly progress report for contract DAAK70-77-C-0248, Prototype Automatic Target Screener. The objective of the effort is to design an automatic target screener to be used with thermal imaging systems employing common module components.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

402 349

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION I

INTRODUCTION AND SUMMARY

This is the fourth quarterly technical progress report for contract number DAAK70-77-C-0248, Prototype Automatic Target Screener (PATS). The first two quarterly reports documented the Phase I design study. The third quarterly report included a description of the final target classifier design for the target data base currently available and the results of the hardware and software system design tasks. Also included were detailed designs of two of the nine functional-level subsystems required. This report includes the design details of the remainder of the subsystems with the exception of the sync and timing circuitry, which has not yet been finalized, interval generation and the symbol generation hardware, which will be designed at a later date. This report covers the period from 1 July to 30 September 1978.

The program objective is to produce a design for an automatic target screener. The screener will reduce the task loading on the thermal imager operator by detecting and recognizing a limited set of high-priority targets at ranges comparable to or greater than those for an unassisted observer. A second objective is to provide enhancement of the video presentation to the operator. The image enhancement includes (1) automatic gain/brightness control to relieve the operator of the necessity to continually adjust the display gain and brightness controls, and (2) dc restoration to eliminate artifacts resulting from ac coupling of the infrared (IR) detectors.

The image enhancement portion of PATS will consist of circuitry to operate on the Common Module FLIR (MODFLIR) video output signal. The circuitry will provide global gain and bias control in the form of feedback to the

MODFLIR to maintain the signal within the dynamic range of the electro-optical multiplexer.

Image enhancement will also include local area gain and brightness control to enhance local variations of contrast and compress the overall scene dynamic range to match that of the display. This circuitry has been completed and examples of its performance on videotaped thermal image data were included, along with the circuit description, in the first quarterly report.

The third image enhancement circuit is for dc restoration to eliminate the streaking associated with loss of line-to-line correlation on the displayed image because of the ac coupling of the detector channels.

This report consists of five sections. Section II describes the results of the DC Restore and Global Gain and Bias Control circuit design and breadboard test results. These circuits are part of the Image Enhancement subsystem, along with the Local Area Gain and Brightness Control Circuit, which has been previously reported. Section III describes the detailed hardware design accomplished during the reporting period. Section IV describes the software design and coding effort to date. Section V describes the planned effort for the next three-month period.

SECTION II

IMAGE ENHANCEMENT

The Image Enhancement subsystem provides three functions: local area gain and brightness control (LAGBC), dc restoration, and automatic (global) gain and brightness control. Algorithms were developed and circuits have been designed and breadboards built and tested for all of these functions. The LAGBC algorithm and breadboard were described in the first and second quarterly reports. Synthetic dc restoration and automatic global gain and bias control results are described below.

SYNTHETIC DC RESTORATION

The algorithm and implementation for the all-analog method of synthetic dc restoration for the common module FLIR was discussed in a previous report.[1] This section presents the results of tests run on a prototype version of the unit as well as problems encountered. Tests run using local area gain/brightness control after and without synthetic dc restore were also made and are reported.

Results Using DC Restore

The test image used was a soldering iron extended horizontally one-third of the way into the FLIR field of view resulting in a black streak the other

---

[1] D. E. Soland, et al., "Quarterly Progress Report, Prototype Automatic Target Screener," Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, Minn., January 15, 1978.

two-thirds of the field of view. A drawing of the test image is shown in Figure 1. The iron was focused to give the sharpest vertical edges possible. Oscilloscope traces of one field of FLIR video before and after dc restoration is shown in Figure 2. The input video is the bottom trace and the dc restored version is at the top. As can be seen in the input video, the background drops in the region of the target. In the dc restored version 2a, the background is nearly flat, as it should be. The background remained nearly flattened for all values of gain and bias adjustments on the FLIR. When the polarity switch on the FLIR was changed, the waveforms in Figure 2b resulted. Note that the background is not as flattened. This is due to a gain error in computing the time during which the difference video is negative, which can be corrected.

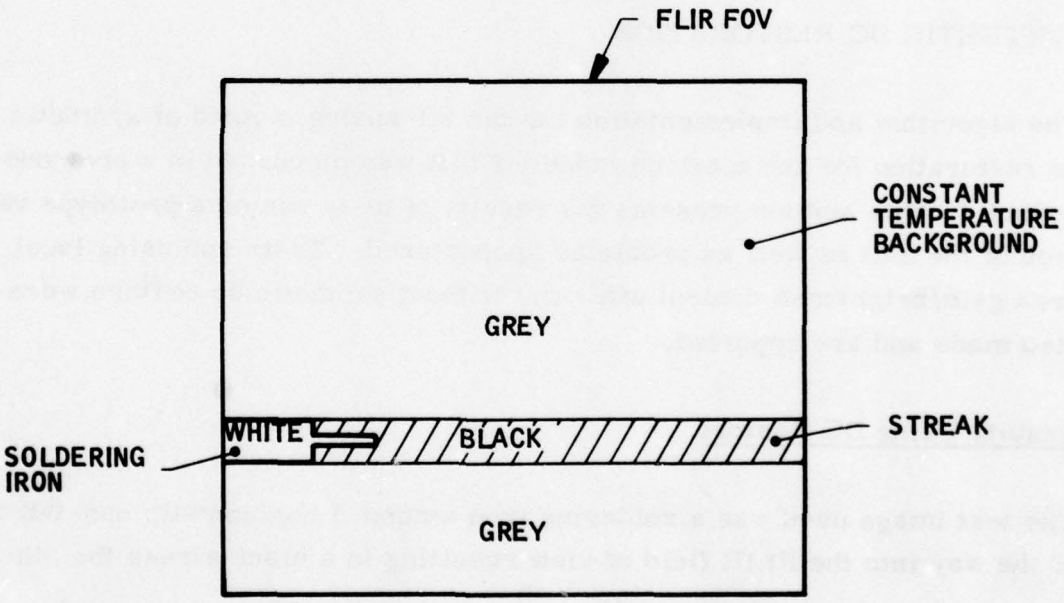


Figure 1. D-C Restore Test Image

Problems which occurred in testing dc restore were its high sensitivity to gain mismatch between the input video and the delay line output video, and to offset voltages in the amplifiers. These errors are not critical if differences between the background regions from line-to-line are large (>50mV), but for the scenes tested, the background was smoothed in the vertical direction, resulting in small line-to-line differences (<10mV at times), thus requiring nulling offsets and "tweaking" gain potentiometers. Another problem was the very high amount of noise in the FLIR video. This caused frame-to-frame variation in the dc restore correction voltage, causing flicker in the restored regions on the output of synthetic dc restore. This is a difficult problem to solve but may be reduced by modifying the T_+ and T_- computation section [2] by using integrators to compute both values.

Results Using Both DC Restore and LAGBC

Since synthetic dc restoration works best on sharp vertical edges in the background, combining it with local area gain/brightness control (LAGBC) may improve its performance on smooth vertical edges. The LAGBC will reduce the contrast between large low-frequency areas but enhance vertical edges between the regions. However, if the vertical edges are already smooth as in the test pattern in Figure 2, the edges won't be enhanced much and the contrast between the grey background and black streak will be decreased. This is shown in Figure 3a with the input at the bottom. The background is smoother after LAGBC. If dc restore and LAGBC are both used the result should be first an elimination of sharp vertical edges in the background by dc restore and then vertical smoothing of the background by LAGBC. This is shown in Figure 3b. As can be seen, the resulting waveform after both functions has a smooth background.

---

[2] Ibid, p. 23

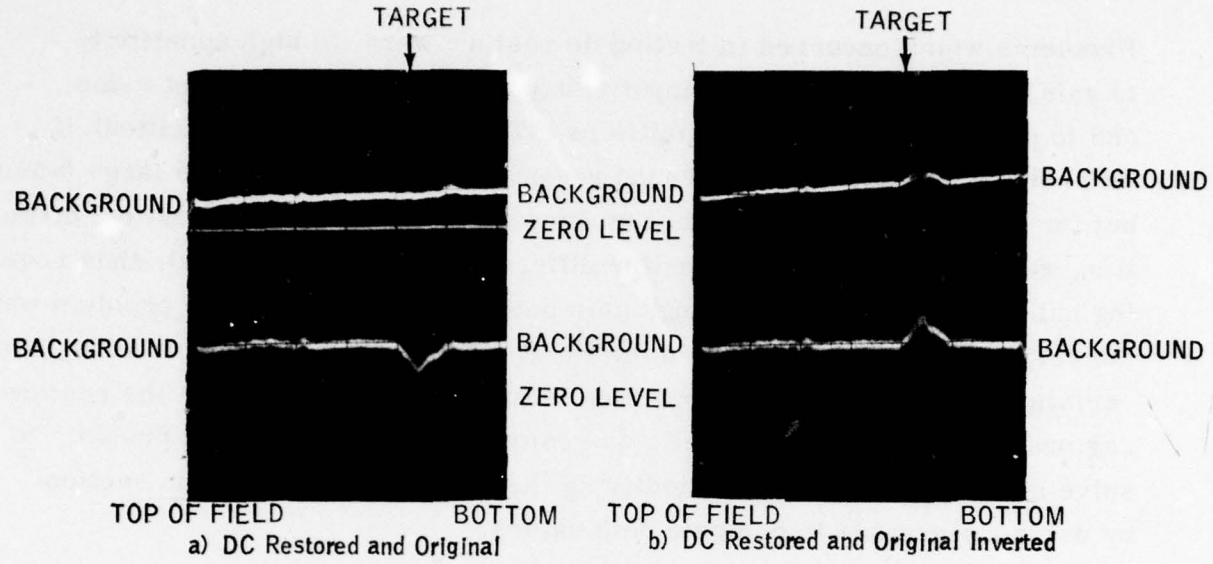


Figure 2. D-C Restore Test Results

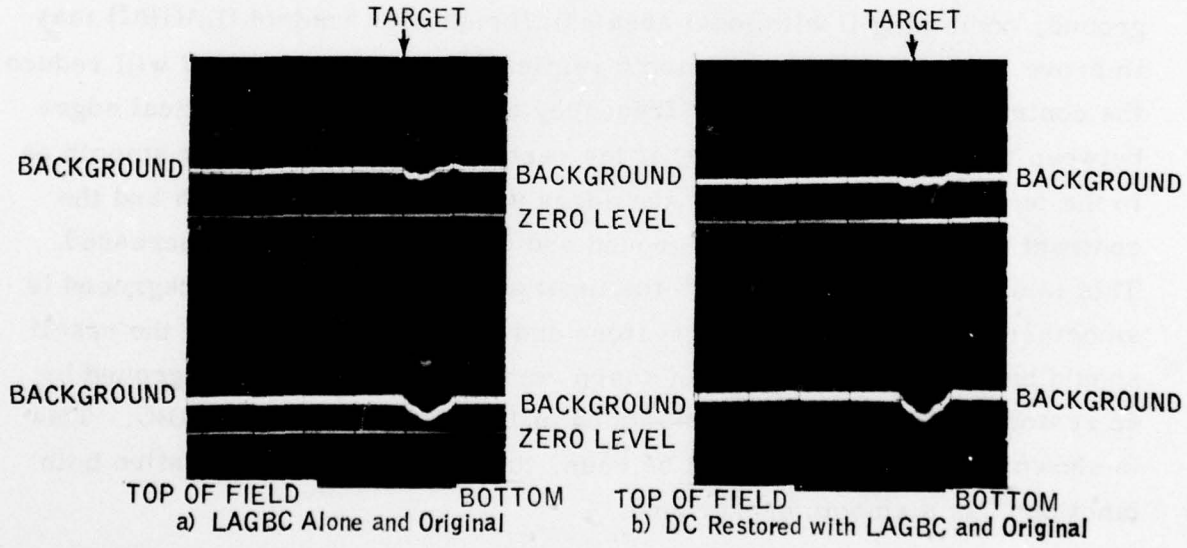


Figure 3. LAGBC and D-C Restore Test Results

AUTOMATIC GLOBAL GAIN AND BIAS CONTROL

The algorithm and implementation for the automation of FLIR gain and bias controls was discussed in a previous quarterly report.[3] In this report, the results of using the algorithm are discussed. Techniques for speeding up the gain updating will also be briefly discussed.

Threshold Setting

The setting for the video saturation thresholds is first examined. One horizontal line of the FLIR video voltage waveform is shown in Figure 4. The black reference in the blanking pulse region is always 0 volts. The active video ranges from 0 to 0.4 volt. The sync pulse is a negative voltage. The average value or d-c component of the active video (excluding blanking regions) is always about 0.2 volt for a static scene and is independent of the gain and bias control of the FLIR. The upper and lower vidicon saturation voltages, V_u and V_l respectively, are bias dependent. When the bias changes, these thresholds must also change since the bias or d-c component of the active video is lost due to a-c coupling after the vidicon*. If the bias is available in the output video, the values of V_u and V_l will always be fixed. With the bias fixed at half the values of vidicon saturation, the threshold voltages were experimentally found to be:

$$V_u = 0.3 \text{ volt}$$

$$V_l = 0.1 \text{ volt}$$

---

[3] Ibid, pages 45-47.

* This may just be a property of this particular FLIR because the FLIR documentation states that the video is clamped to the vidicon black level during electron beam retrace which should bias restore the video. This is not the case in the FLIR that Honeywell is currently testing.

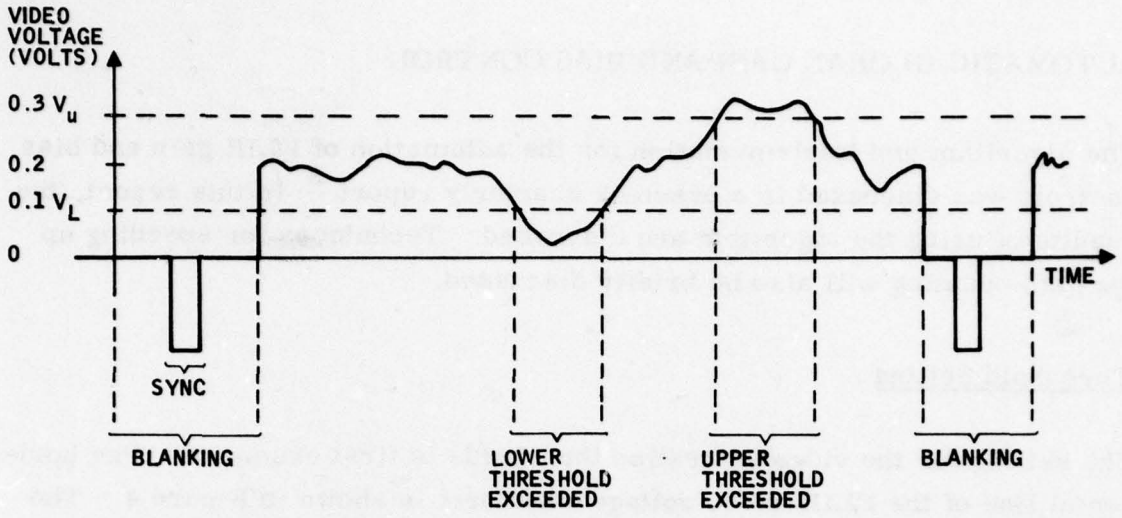


Figure 4. One FLIR Horizontal Video Line

Test Results

The automated gain/bias control unit was tested by moving a soldering iron in and out of the field of view of the FLIR. Only the gain was varied; the bias was controlled by the microprocessor

The constants in the microprocessor software which changed since the last reporting period[4] are:

K_o = 0.008N (N = pixels/field)

K_1 = 0.01N

K_2 = 0.02N

ΔG_I = 6 percent (of total gain range)

ΔG_D = 8 percent (of total gain range)

---

[4] D. E Soland et al, "Quarterly Progress Report, Prototype Automatic Target Screener," Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, Minn., 15 June 1978.

The number of fields of threshold exceedance data sampled was reduced from eight fields to four The constants were changed after experimenting with the test pattern, and were found to work well.

The soldering iron was first left out of the field of view of the FLIR, leaving a fixed temperature background scene. For this situation, the gain was set to maximum in order to see small detail (a low contrast scene). The video waveform from top to bottom of the scene is shown on an oscilloscope trace in Figure 5a. When the soldering iron was moved into the scene, the waveform in Figure 5b resulted if the automatic gain control was not being used (manual operation). The waveform in Figure 5b after automatic gain control is shown in Figure 5c. The time to set the gain correctly was about 5 seconds. The soldering iron was removed from the field of view and with manual gain set to the same as the automatic gain in Figure 5c, the waveform in Figure 5d resulted. With automatic gain, Figure 5a was once again obtained. Again gain update took nearly 5 seconds.

In the scenes shown in Figure 5, the gain varied from minimum to maximum and were worst cases as far as the time to update Where saturating targets move in and out of the scene slowly with respect to the FLIR field of view, gain control will take less time.

Alternate techniques for efficiently finding the "optimal" gain have also been explored. The reason for using a small fixed-gain increment or decrement was to keep the gain system critically damped to eliminate overshoot and maintain stability. The overshoot probably is not a critical factor if speed of update is important. Underdamping the system can be accomplished by decreasing the time between gain update; i. e., don't wait for the FLIR electronics to completely stabilize with a new gain value. This was tested by decreasing the wait time for the FLIR from 30 fields to 10 fields. The results of the test were a reduction in maximum gain update time to 2 seconds, but overshoot was apparent on some of the scenes.

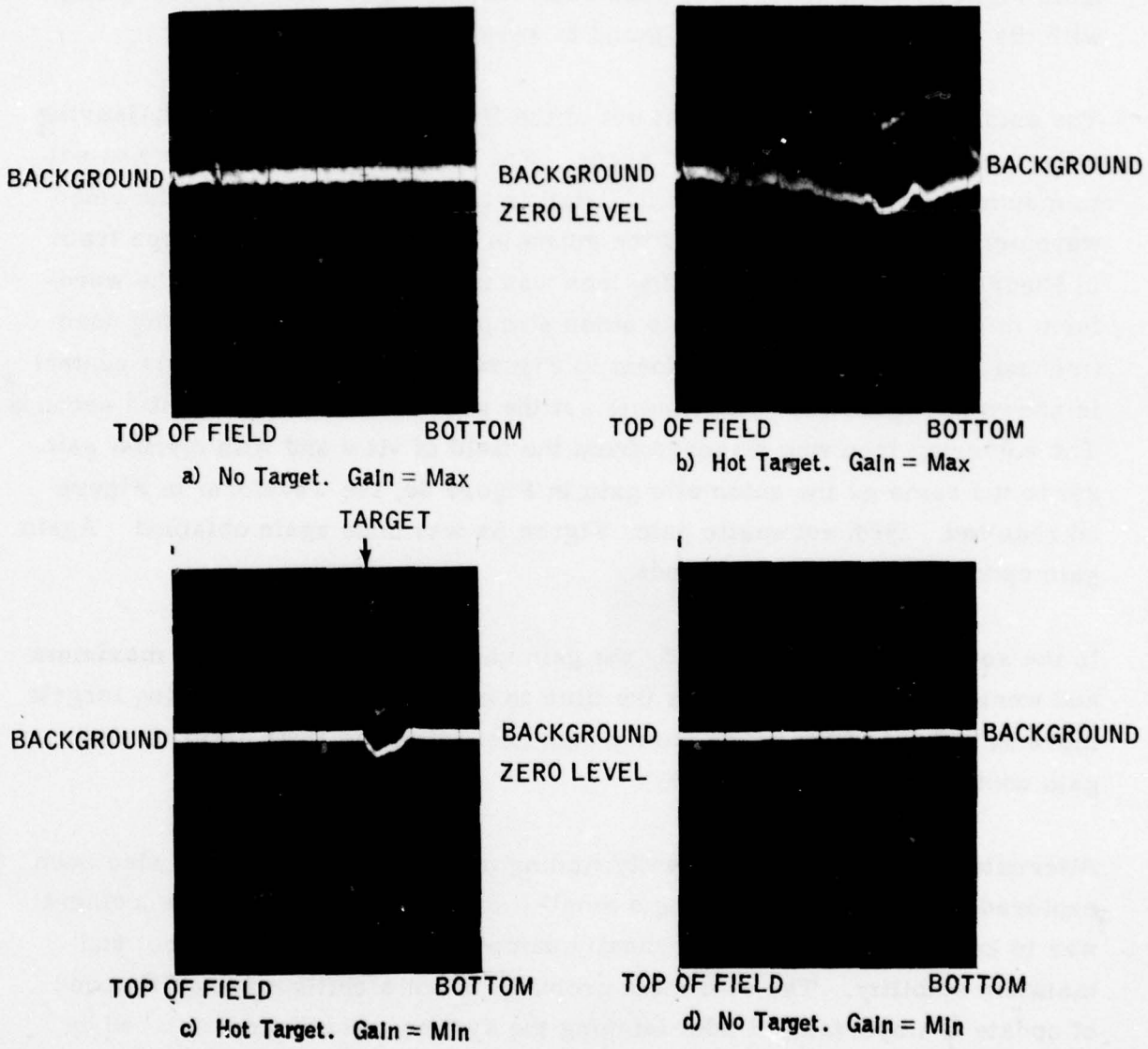


Figure 5. Test Results for Global Gain and Brightness Contrast

Other techniques tested for speeding up gain control were to let ΔG_D double if the threshold exceedance (N_T) exceeded 0.05N and double ΔG_I if $N_T <$ 0.004N. This method decreased the update speed to a maximum of three seconds but exhibited overshoot and seemed to oscillate at times.

SECTION III

HARDWARE DESIGN

This section describes those design tasks that were completed during this reporting period. In the previous quarterly report (Section IV) the PATS hardware design tasks were broken down into the following subparts:

1. Image Enhancement
2. Edge Signal
3. Bright Signal
4. Interval Generation
5. CPU 1
6. Memory 2 (Intensity Information)
7. CPU 2
8. Symbol Generation
9. Sync and Timing

In the previous report, the design task for edge signal and sync and timing were reported. CPU 2 is an off-the-shelf component system and is currently being assembled. Symbol generation design and interval design will be taking place later. The image-enhancement circuitry has been breadboarded and minor design changes will be taking place in order to incorporate it in the PATS hardware.

Three design tasks have been completed during the current reporting period:

1. Bright Signal Generation
2. Memory No. 2 which includes a 256 x 512 x 6 bit memory, A/D conversion, background estimate, and intensity summation
3. Digital Processing Subsystem (CPU 1)

The bright signal-generation circuit is being built, and part of Memory No. 2 is being breadboarded with final build to take place with printed circuit boards after initial checkout. CPU No. 1 design is nearly complete and should be ready for build by the end of October.

BRIGHT SIGNAL GENERATION

The bright signal-generation block diagram is shown in Figure 6. This unit is the first that is used in the preprocessing for feature extraction in PATS.

The video data is initially low-pass filtered at 5 megahertz for noise limiting and anti-aliasing. During periods of blanking, the input video is clamped at zero volts reference. This video goes to the rest of the PATS hardware which includes the edge circuitry, the intensity memory, A/D circuitry and the bright circuitry.

The background filter consists of a two-dimensional filter. In the horizontal direction, a 800-KHz low-pass filter is inserted in the video line followed by a horizontal scan line delay and an adjustable time constant related to the number of lines. This vertical filter consists of taking a fraction (β) of the previous background and adding to it ($1 - \beta$) of the incoming video.

The output of the line delay is low-pass filtered (5 MHz) to get rid of the clock noise generated in the CCD. The output of the filter is an estimate of the background, pixel by pixel. If it is determined that the incoming video contains data that is target-like in nature, then a switch holds the incoming video. The signal that performs this control is the interval signal, which is generated elsewhere in the PATS hardware.

The background estimate is buffered before it goes to the A/D section in the PATS hardware (the background estimate derivation is described in the section relative to Memory No. 2). The background estimate is subtracted from

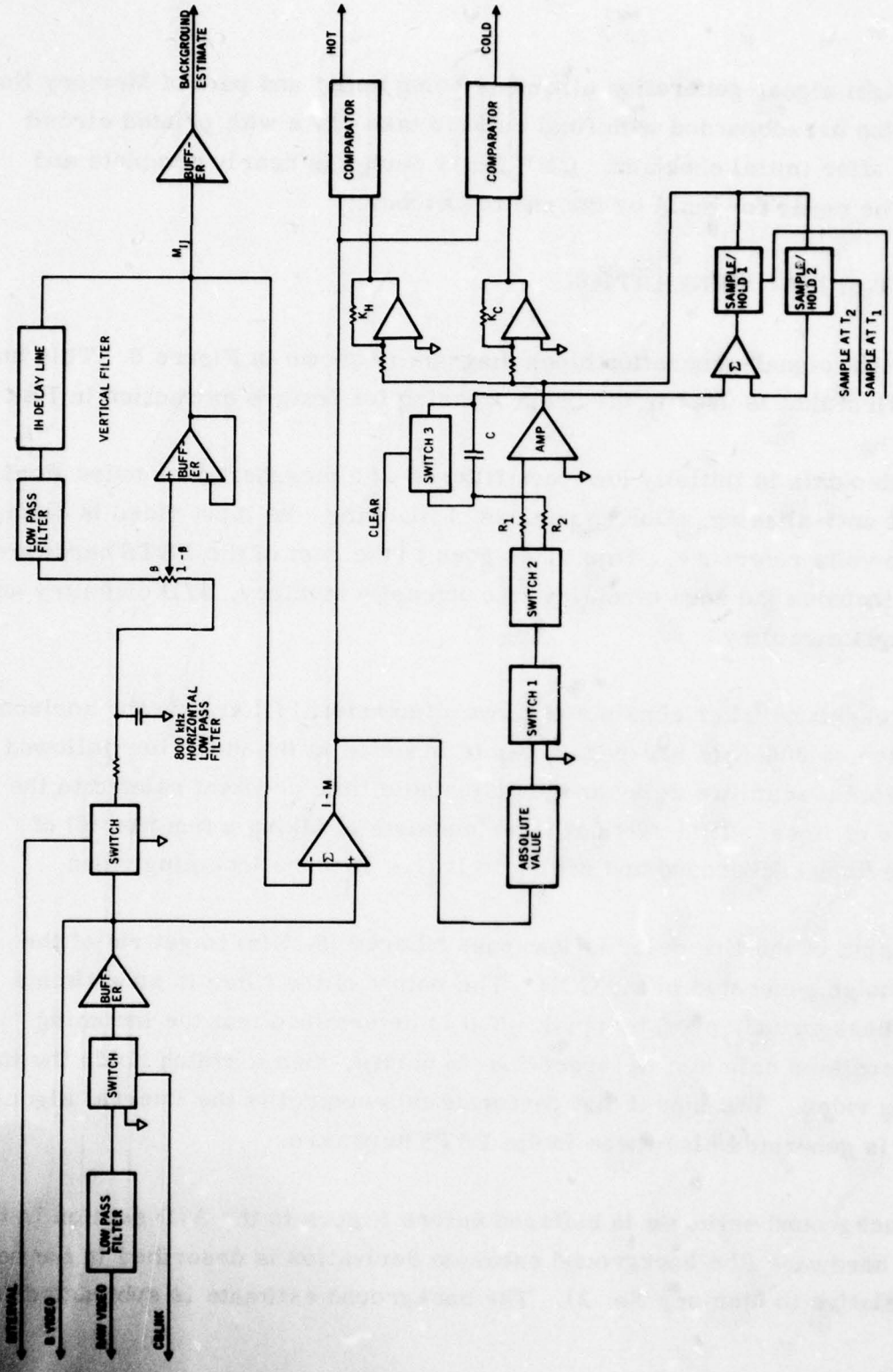


Figure 6. Bright Signal Generation

the incoming video. What remains is the higher frequency variations in the data since the background filter has taken out the low-frequency variations.

To get an estimate of the variation, the data is passed through an absolute value circuit, the average absolute variation is determined and a threshold is calculated. The threshold is based upon the present value and several previous values in much the same way as described in the edge threshold in the previous quarterly report.

The thresholds for the hot and cold signals have different multiplier coefficients. These will not be adjustable by the operator but the values chosen for the multiplier can enable one to allow more cold objects than hot objects and vice versa by changing respective coefficients. For initial testing, these coefficients will be set equal and not biased toward a hot or cold object.

The comparator will compare the higher frequency varying data with the thresholds. The HOT signal generated will be all those values above the hot threshold and COLD signal will be all those values below the cold threshold. The HOT and COLD signals go to the interval generation and first-level feature hardware.

INTENSITY MEMORY (MEMORY NO. 2)

Memory No. 2 is a digital video field store memory. Digital data is loaded sequentially into the memory from the analog-to-digital converter unit (ADCU). The memory can currently store 256 video lines of 512 samples per line and 6 bits of digitizing resolution. The number of digitized lines is expandable to 512 and the resolution can be expanded to 8 bits. The memory data can be read out by CPU No. 1 or CPU No. 2 randomly or sequentially. A capability can also be added to do a video read for displaying memory data on a video monitor. The memory can operate at 20 mega-words (6-8 bits) per second sequentially and averages 2.5 mega-words per

second in random mode. The ability to combine sequential and random-access modes when reading the memory allows much higher average word rates than obtained in just random access mode. The memory is constructed using 16 kilo-words by 1 bit dynamic random access memories (DRAMs) with 375 nsec read/write cycle times.

Memory No. 2 Design

A basic block diagram of Memory No. 2 is shown in Figure 7. Digitized video or diagnostic data from Multiplexer No. 2 on the ADCU is sequentially entered into the memory section. Digitizing of one field-of-video data is controlled by the Memory No. 2 timing and control section. When CPU No. 1 issues a sample field instruction, the memory is loaded with one field of data. After loading the memory, the memory is put in a read-only mode which is controlled by CPU No. 1 or CPU No. 2. The digitized data is read out of memory on command from the CPUs which furnish a memory address of the location to be read. Data is read onto the CPU data bus. Timing clocks to Memory No. 2 are furnished by the Sync and Timing Unit (STU).

Memory No. 2 Bit Plane -- Memory No. 2 uses a bit plane structure which has very fast sequential access times and has resolution expansion capabilities. Each bit plane stores one bit of information as seen in Figure 7. Currently, Memory No. 2 has 6 bit planes, expandable to 8. Each bit plane has 512 x 256 single-bit words. The bit plane memory size can be doubled to a 512 x 512 bit plane. A block diagram of the bit plane used in Memory No. 2 is shown in Figure 8. Eight 16K x 1 DRAMS form the 512 x 256 word memory. Bit N (N = 0-5) of the digitized video from the ADCU is the input to an 8-bit serial-to-parallel shift register. The shift register is clocked by SRC LKH which is the same as the LSUMH clock on the ADCU. When 8 bits are shifted into the shift register, the contents are loaded into an input pipeline latch by the INC LKH signal from the Memory No. 2 timing and control unit. After the 8-bit input pipeline latch is loaded by INC LKH a memory

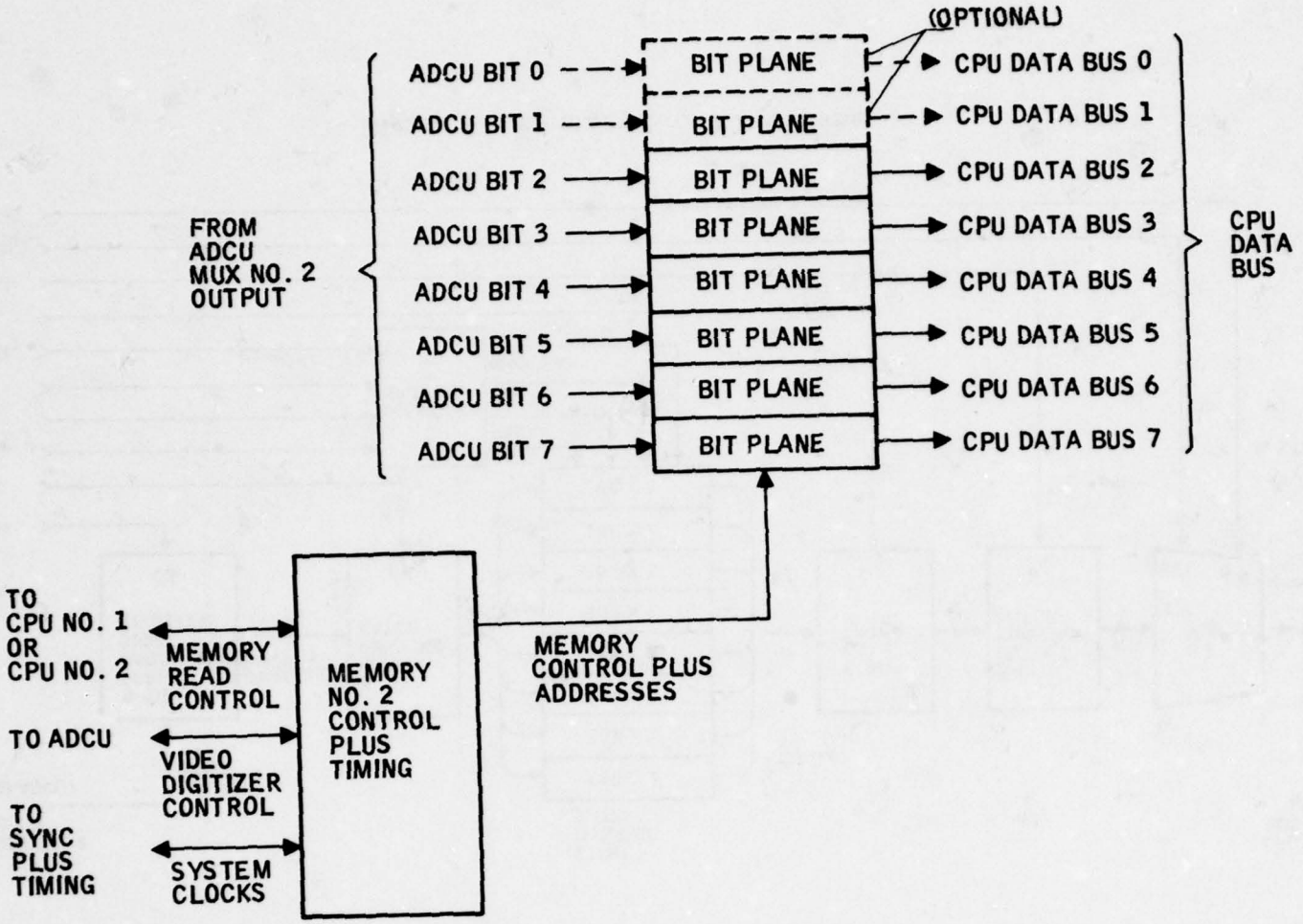


Figure 7. Memory No. 2 Block Design

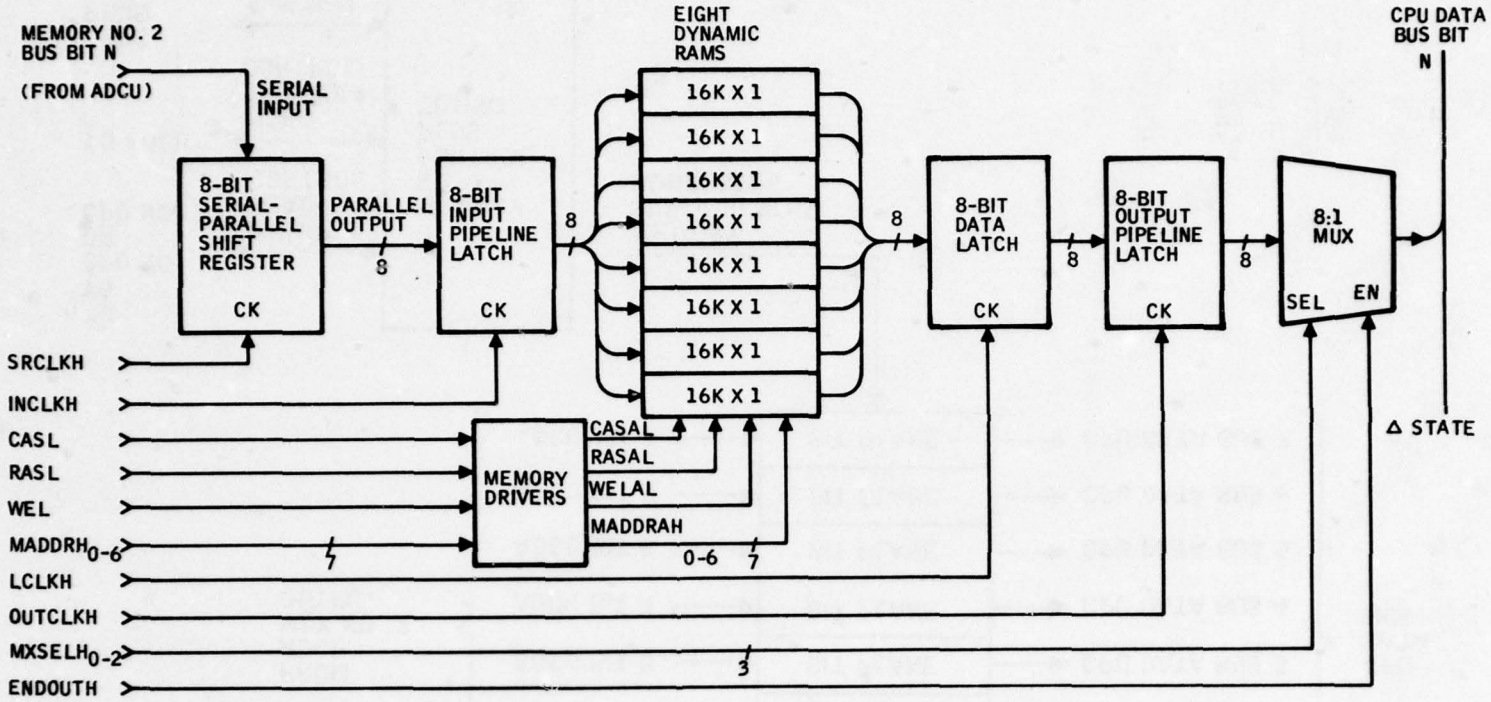


Figure 8. Memory No. 2 Bit Plane

cycle is issued which writes eight consecutive single-bit words. At the same time the write cycle occurs, the shift register can be loading the next eight consecutive words of digitized data. This gives an effective sequential cycle time of one-eighth of the actual 16K x 1 DRAM cycle time.

The DRAM memory timing signals (RASL, CASL, WEL) and the multiplexed memory address (MADDR(0-6)H) are from the Memory No. 2 timing and control. The read technique on the bit plane is to first read eight consecutive single-bit words from the memory block and latch them in the 8-bit data latch with the LCLKH signal (the DRAMS have no output latches). The output of the data latch goes into an 8-bit output pipeline latch controlled by OUTCLKH and then to an 8:1 multiplexer onto the tri-state CPU data bus bit N (same as N from ADCU). The multiplexer is controlled by the three least significant bits of the memory address (MXSEL(0-2)H) and is enabled by the ENDOUTH signal. The output pipeline latch allows the memory to access the next eight word block while reading the current eight words out through the multiplexer. This allows the sequential access output data rate to be as fast as the input sequential data rate.

Fujitsu 8116H DRAMS with 375 nsec read/write time are being used for the memory, but many of the other industry DRAMS can directly substitute.* Standard 74LS series TTL makes up the remaining electronics.

Memory No. 2 Control and Timing Unit -- The Memory No. 2 control and timing unit (MCTU2) shown in Figure 7 interfaces to the two CPUs, controls the video digitizing and sets up controls, timing and addresses to the bit plane memories. The MCTU2 operates in one of two modes: (1) storage of one field of digitized video data into memory, or (2) reading memory when commanded by and at addresses furnished by either of the two CPUs.

*The TMS4164 64K x 1 DRAM will directly substitute allowing a quadruple size in memory with no physical size increase.

A block diagram of the MTCU2 is shown in Figure 9. When a field is being digitized, the CPUs cannot access the memory until the end of the field. When field digitizing is done, the CPU communicates with the MTCU2 over the CPU address bus (ADDRH 0-15) through the CPU Interface and Control Decode Section. The control word format from the CPUs is shown in Table 1. When bit 15 (ADDRI5H) is a 1, Memory No. 2 is selected. Address loading into the MTCU2 is controlled by bits 13 and 14. Addressing of the memory is in an X, Y format where X is the horizontal (pixel in a video line) and Y is the vertical (the video line) address in the 512 (X) by 256 (Y) word memory bit plane. Memory read cycles are initiated by bit 12. The X and Y addresses are incremented automatically to permit sequential access according to bit 11. Bit 10 can be set to use the output pipelined latch on the bit planes allowing high-speed access sequentially. Bits 0-8 are the X or Y address data.

The CPUs inform the MTCU2 that valid data is on the address bus by setting VALIDL low as shown in Figure 10. As long as VALIDL is low and ADDR15H is 1, the CPU data bus (DATA 0-5 H) is dedicated to the MTCU2 (ENDOUTH is high).

The VALIDL signal remains low until the READYL signal goes low and is sent to the CPUs by the MTCU2. The time from VALIDL going low until READYL goes low is dependent on the operations requested in bits 10-14 of the CPU address bus control word. If a load with no memory read cycle is requested, t_{cy} is much less than a CPU cycle ($\simeq$50 nsec). When a memory read cycle is requested, t_{cy} is highly variable due to the periodic refresh cycle "stealing" and the bit planes reading eight consecutive words at a rate much higher than the DRAM cycle rate. The time for t_{cy} can range from 50-750 nsec, but will average 50 nsec sequentially (short cycles) and 375 nsec when access is highly random (long cycles). The CPU Interface and Control Decode latches bits 10-14 when VALIDL goes low, generating the signals OUTPH (bit 10), AUTOYH (bit 11), MCYH (bit 12), LDYH (bit 13)

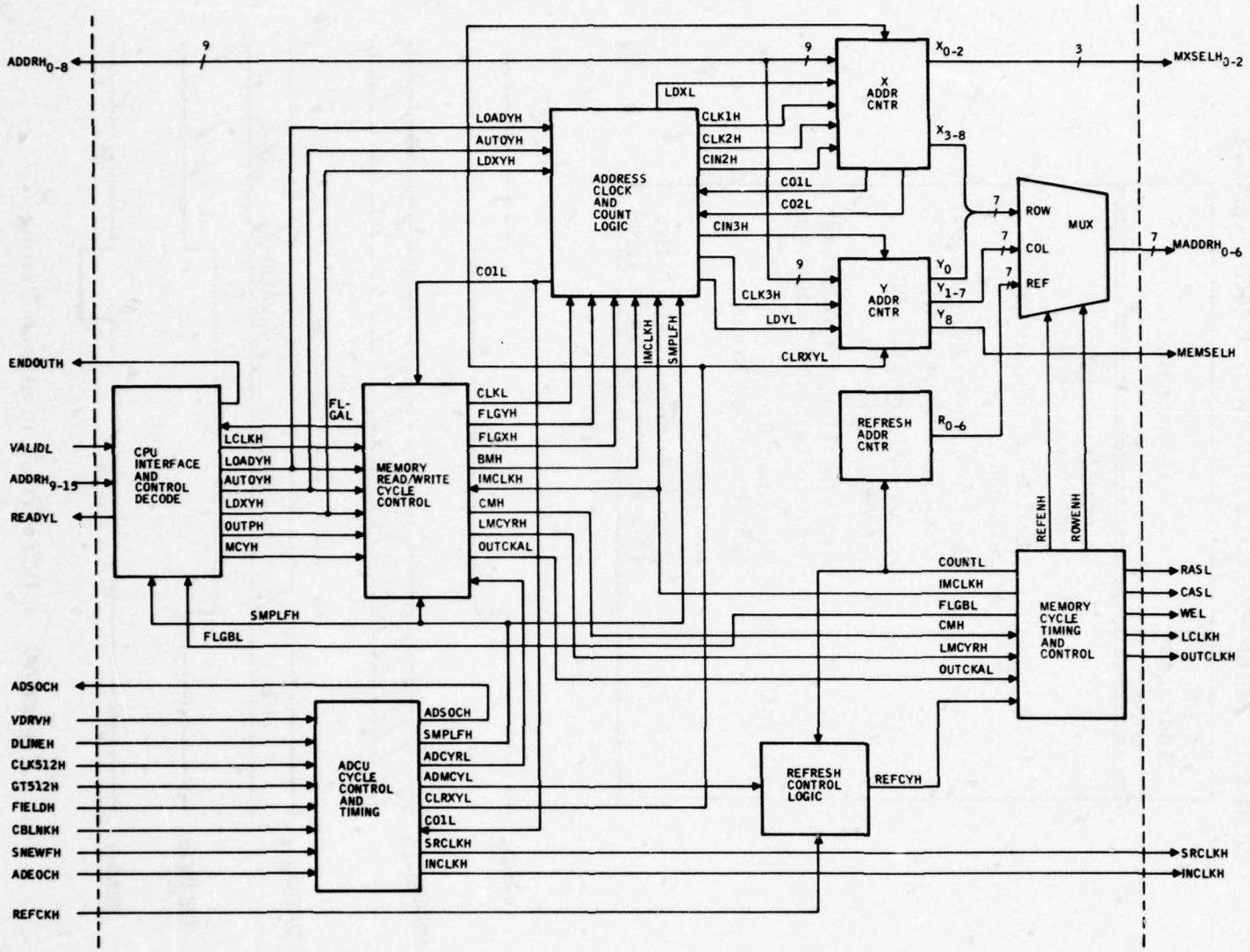


Figure 9. Memory No. 2 Timing and Control Unit (MTCU2)

TABLE 1. MEMORY NO. 2 CONTROL WORD FORMAT

CPU Address Bit(s)	Description
15	1 Select Memory No. 2 0 Select Memory No. 1
14	1 Load X or Y Address 0 No Load
13	1 Load Y Address (Bit 14 = 1) 0 Load X Address (Bit 14 = 1)
12	1 Do Memory Cycle 0 No Memory Cycle
11	1 Auto-Increment Y Address 0 Auto-Increment X Address
10	1 Pipelined Read 0 Non-Pipelined Read
9	Not Used
8-0	X or Y Address Data if Bit 14 = 1, Otherwise Not Used

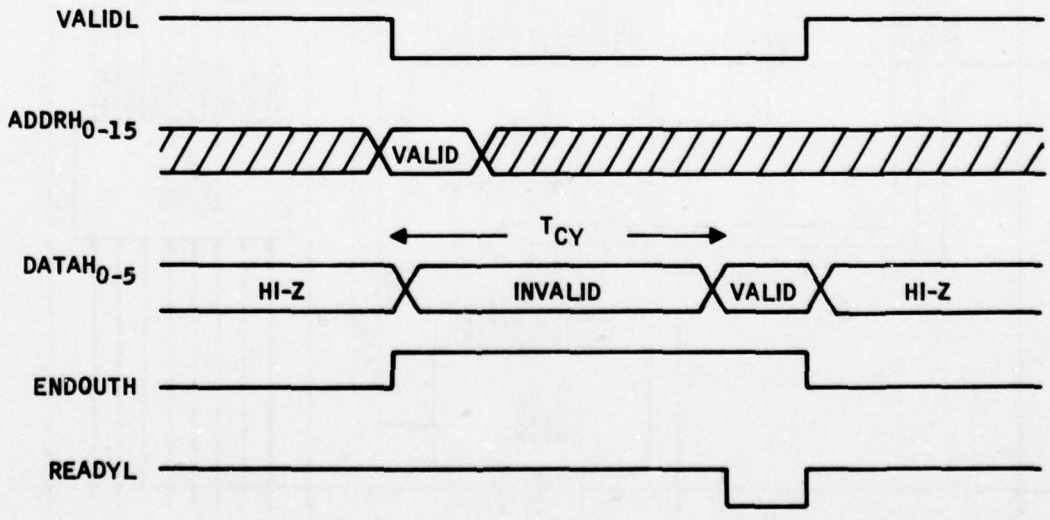


Figure 10. CPU/MTCU2 Interface Timing

and LDXYH (bit 14) plus a clock signal (LCLKH) which when high means the above latched values are not stable. The READYL signal is set low when either of two flag inputs (FLGAL, FLGBL) goes low and is cleared when VALIDL is high. The CPU address bus latch is cleared when a field is being sampled (SMPLF is high) by the ADCU Cycle Control and Timing section.

The CPU Interface section furnishes inputs to the Memory Read/Write Cycle Control and Address Clock and Count Logic sections. The Memory Read/Write Cycle Control generates a clock to the address counters and cycle requests to the Memory Cycle Timing and Control Section. There are two types of memory read cycles - long cycles and short cycles. Long cycles require reading an 8-bit word from the DRAMS on the bit planes while short cyles access only the data currently latched on the bit plane. These two cycles result from the addressing structure of the bit planes and that the least significant 3-bits of the X address (X_0 - X_2) control the multiplexer on the bit plane ($MXSELH_{0-2}$). Long cycles result when any Y address bit or X address bits 3-8 change from a previous cycle, whether by incrementing the address or loading a new one. Short cycles occur if only X address bits 0-2 change. This allows for highly efficient sequential and random access in the X direction. Long cycles always occur if a load address was made (LDXYH high) or a carry out occurred in the lower 3-bits of the X address (CO1L low). A long cycle issues a long memory cycle request (LMCYRH) to the Memory Cycle Timing section. A clear long memory cycle (IMCLKH) is returned when the DRAMs are read and a flag (FLGAL) is generated back to the CPU. In short cycles as well as long, a clock is issued to the address clock and count logic to increment count values. A short cycle sets the proper address ($MXSELH_{02}$) and returns FLGAL without accessing the DRAMs. The Memory Read/Write Cycle Control also generates flags for the output pipeline option (BMH, CMH, and OUTCKAL) and flags indicating whether an X address (FLGXL) or Y address (FLGYL) was loaded since the last long memory cycle.

A description of the addressing sections will be given. The MTCU2 generates all addresses for the memory bit planes; addresses are common to all bit planes. The DRAMs themselves require 14 address bits multiplexed into two 7-bit row and column addresses. All row addresses must be refreshed (or read) every 2 msecs or the memory will drop bits. The 7-bit address to the memory (MADDR(0-6)H) is one of three addresses: (1) the refresh address, a (2) row, or (3) column address from the X, Y address latch/counters either set-up internally by the MTCU2 or loaded by one of the CPUs. The X and Y address latch/counters are each 9 bits wide. The X address accesses horizontally (along video lines) and the Y address vertically (the video line) in the memory. The least significant bits ($X_0 - X_2$) of the X address control the 8:1 output multiplexer on each bit plane ($MXSELH_{0-2}$). The X address bits 3-8 ($X_3 - X_8$) and Y address bit zero (Y_0) make up the 7-bit row address to the DRAMs. The 7-bit column address is $Y_1 - Y_7$. The most significant bit of the Y address (Y_8) can select another 8 DRAM block in a bit plane to double the memory size; it is currently unused. The refresh address is generated by the 7-bit refresh address counter.

The X, Y addresses can be loaded via the CPU address bus, cleared or incremented. Incrementing can be done on the X address or the Y address. The X address will carry out into the Y address, but Y will not carry into the X address. The X address is broken into two counters; $X_0 - X_2$ forms one and $X_3 - X_8$ the other. The $X_0 - X_2$ counter may or may not carry into $X_3 - X_8$ allowing the $X_0 - X_2$ value to be an independent counter. Clocking, carry and load control are controlled by the Address Clock and Count Logic which receives its inputs from the CPU Interface and Control Decode, the Memory Read/Write Cycle Control, Memory Cycle Timing and Control and the ADCU Cycle Control and timing. The address counters are synchronous counting and loading types. The X address counter is clocked by two clocks, one for $X_0 - X_2$ (CLK1H) and one for $X_3 - X_8$ (CLK2H). When the load X address signal (LDXL) is false, a clocking increments the proper counter. When the load is true, the data on the CPU address bus bits 0-8 ($ADDRL_{0-8}$)

are loaded into all of the X-address. The Y address counter has one clock (CLK3H) and a similar clocking/loading procedure for the load Y-address signal (LDYL). The logical equations for loading and clocking are as follows:

$$LDXH = LOADYL \cdot LDXYH$$

$$LDYH = LOADYH \cdot LDXYH$$

$$CLK1H = CLKH\,(AUTOYL \cdot FLGXL + LDXH)$$

$$CLK2H = IMCLKH\,(SMPLFH + BMH) + CLKH \cdot (SMPLFH + BMH) \cdot (AUTOYL \cdot FLGXL + LDXH)$$

$$CLK3H = IMCLKH\,(SMPLFH + BMH) \cdot (LDYH + FLGXL \cdot LDXL \cdot AUTOYL + FLGYL \cdot AUTOYH)$$

The logic allows the address counters to be used for CPU reads, video writes and video reads.

The ADCU Cycle Control and Timing section uses $X_0 - X_2$ of the address counters as an independent counter and $X_3 - X_8$ and $Y_0 - Y_8$ as a single-address counter. The ADCU Cycle Control and Timing will digitize a frame when the sample new frame (SNEWFH) is issued by the CPUs. Digitization begins when the signal to digitize a line (DLINEH) is enabled (the address counters are cleared prior to this). Digitizing continues until DLINEH is disabled. The start of conversion clock (ADSOCH) is generated by the 512 clock (CLK512) from the sync and timing unit (STU). The address X_{0-2} is incremented by the end of conversion signal (ADEOCH) back from the ADCU. When the carry out from X_{0-2} (CO1L) is true, a long cycle is issued to the bit plane memory by ADCYRL (clocks to the shift register (SRCLKH) and input pipeline latch (INCLKH) are generated by ADEOCH and ADCYRL, respectively).

While digitizing, the CPU interface is disabled (by SMPLFH) as well as the Refresh Control Logic (by ADMCYL) since the memory is automatically refreshed by the high-speed sequential mode writing. When digitizing is finished, refresh is re-enabled and CPU control is once more established.

The refresh control logic generates refresh cycle requests (REFCYH) every 16 μsec or 128 refresh cycles every 2 msec. The refresh request is cleared by the COUNTL signal from Memory cycle timing; COUNTL also increments the refresh address. Refresh is disabled by ADMCYL during field digitization forcing REFCYH low always.

The memory cycle timing and control section generates the proper signals to the DRAMs on the bit planes, generates clocking to the bit plane data and pipeline latches and processes memory read/write and refresh requests. Memory read/write cycle requests (LMCYRH) and refresh cycles (REFCYH) are processed in the order they are received. When a cycle request is recognized, signals are generated to the bit planes and address MUX. The timing is shown in Figure 11. In a normal read cycle, a cycle request is made and REFEN is low on the MUX. The ROWENH signal selects the row address on the MUX ($t_{row} \approx 40$ nsec). When RASL goes low, the DRAM cycle is initiated and the row address is loaded into the DRAMs. After a delay (≈ 20 nsec), ROWENH goes low setting the column address on the MUX. The CASL signal goes low ($t_{rcd} \approx 40$ nsec) loading the column address in the DRAMs. The data from the DRAMs ($DRAMDH_{0-7}$) is valid after a delay ($t_{rcd} \approx 150$ nsec) and is latched in the data latches on the bit planes by LCLKH. A short delay later (≈ 20 nsec) the pipeline latch is clocked by OUTCLKH if CMH is a 1. Next, the increment and read request clear pulse (IMCLKH) is issued followed by an end-of-cycle signal (FLGBL) to the CPU. A refresh cycle sets REFENH high on the MUX, overriding the ROWENH MUX control. The CASL signal remains high during refresh and only COUNTL goes low of the other signals for clocking the refresh counter.

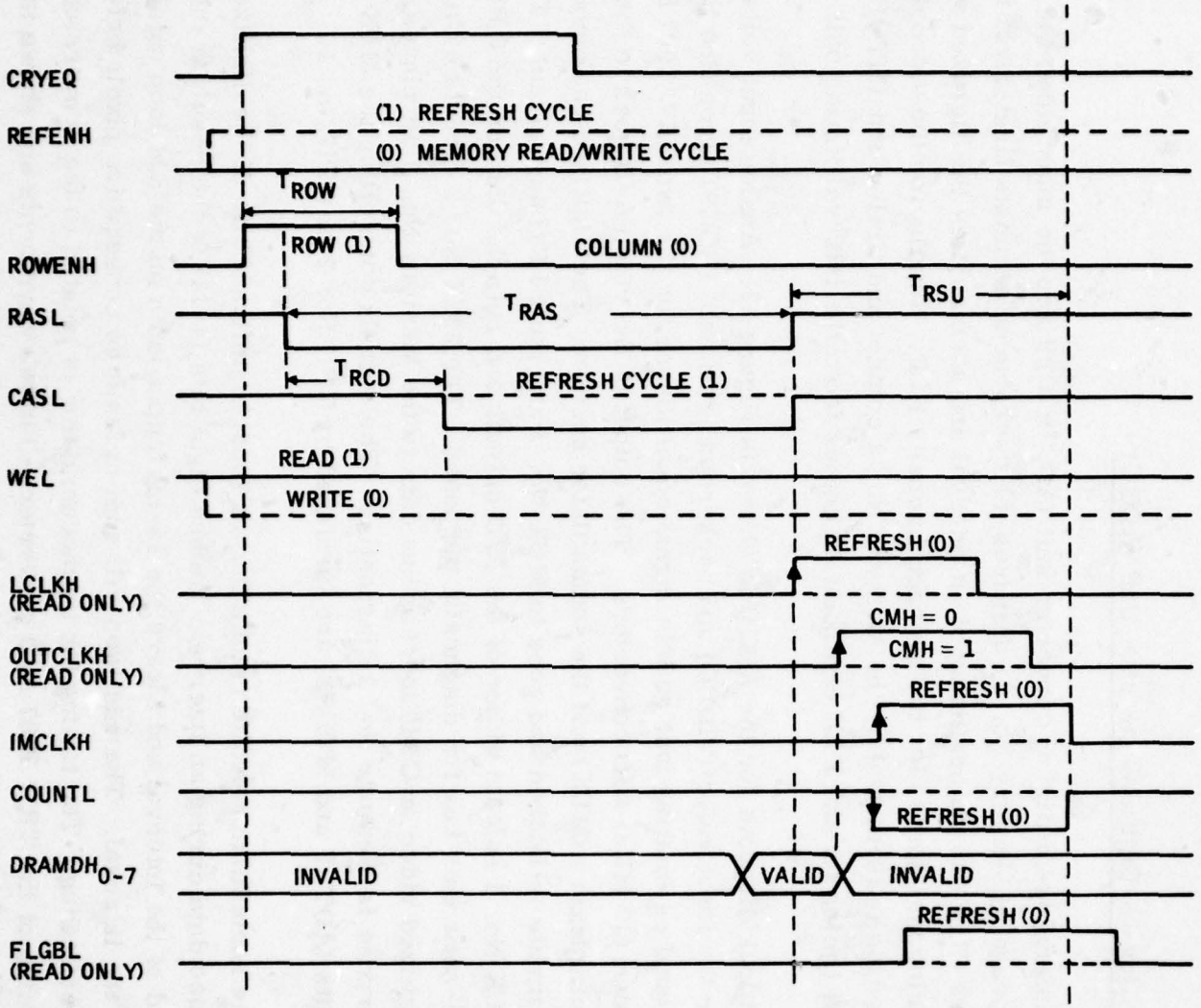


Figure 11. Memory Cycle Timing Diagram (MTCU2)

The MCTU2 is implemented using 74LS and 74S TTL logic. Critical timing is done using passive TTL delay lines.

Analog-to-Digital Converter Unit (ADCU)

The analog-to-digital converter unit (ADCU) digitizes the input video for storage in Memory No. 2, digitizes the background estimate first-level feature (FLF) at the beginning of an interval and accumulates the digitized video during an interval for the average intensity FLF. Timing for the video digitization is furnished by the Memory No. 2 control and timing unit (MTCU2), and timing for the FLF computation comes from the interval timing unit.

A block diagram for the ADCU is shown in Figure 12. Analog input voltages for the input video (VIDEO) and background estimate (BGEST) from the background estimating unit go into signal conditioners for the two TRW 1007 high-speed (30 MHz) A/D converters. The output of the video A/D goes to the two multiplexers (MUX) and the accumulator section. The digitized background estimate is latched and goes to MUX No. 1 and the FLF/Interval unit. The MUX No. 1 selects either of the A/D outputs and puts the data on the CPU tri-state data bus for diagnostic purposes. The MUX No. 2 directs either digitized video or CPU address bus data to the Memory No. 2 bit planes; its purpose is Memory No. 2 diagnostics. The signals controlling the MUX units (ADTST and MTEST) are furnished by CPU No. 2 or CPU No. 1.

The accumulator section sums the six most significant bits of the digitized video data during an interval, latches eight bits (4-11) of the result at the end of the interval and clears the 12-bit output latch on the add loop outside of the interval. The add loop will sum at least 64 consecutive pixels before overflowing. The timing for the accumulator is related to the conversion timing of the TRW 1007 A/D converter. Timing waveforms are shown in Figure 13. The start-of-conversion pulses (ADSOCH) from the Memory No. 2 control unit starts digitizing of the input video. The clocking on the A/D

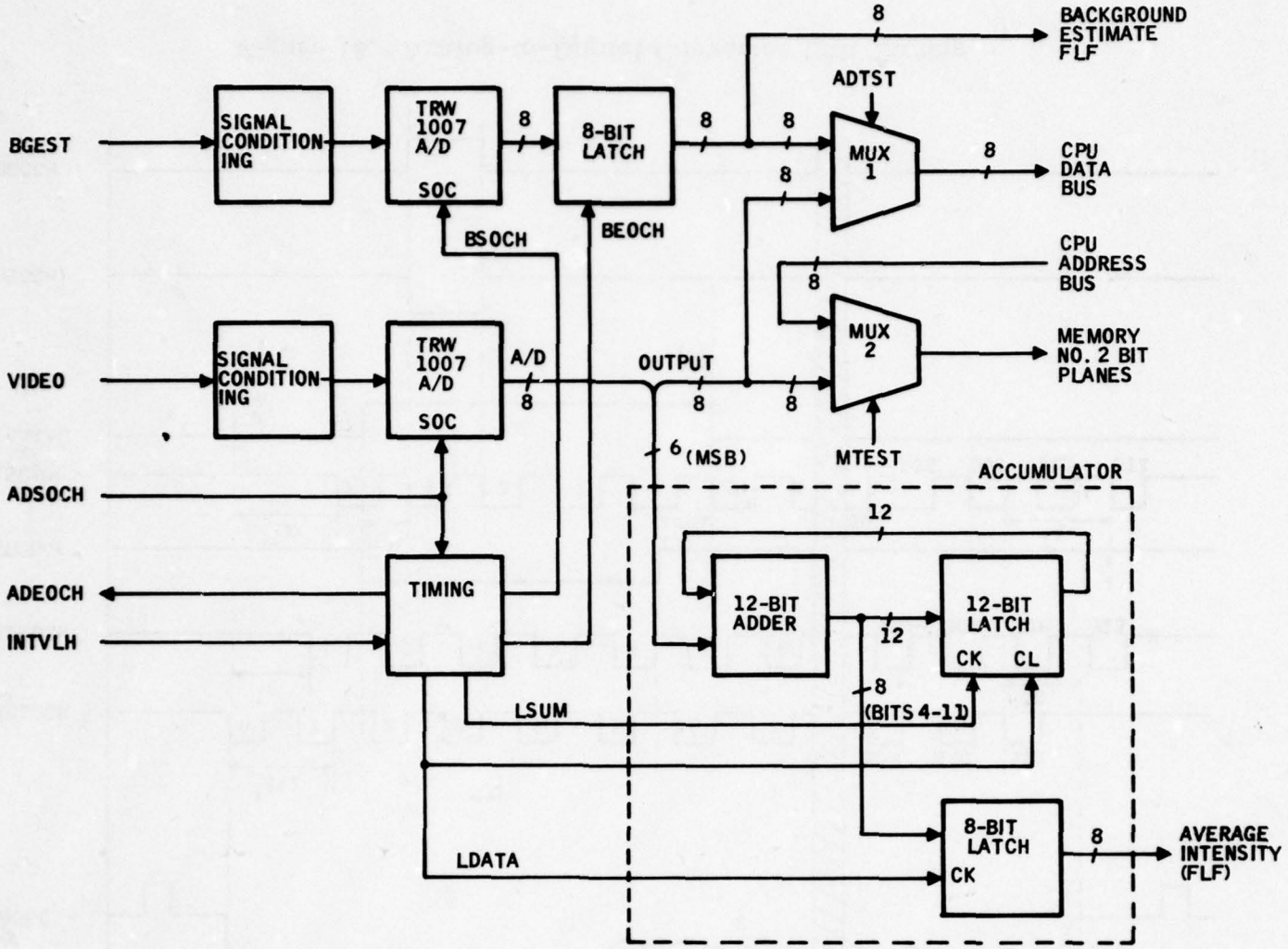


Figure 12. Analog-to-Digital Converter Unit

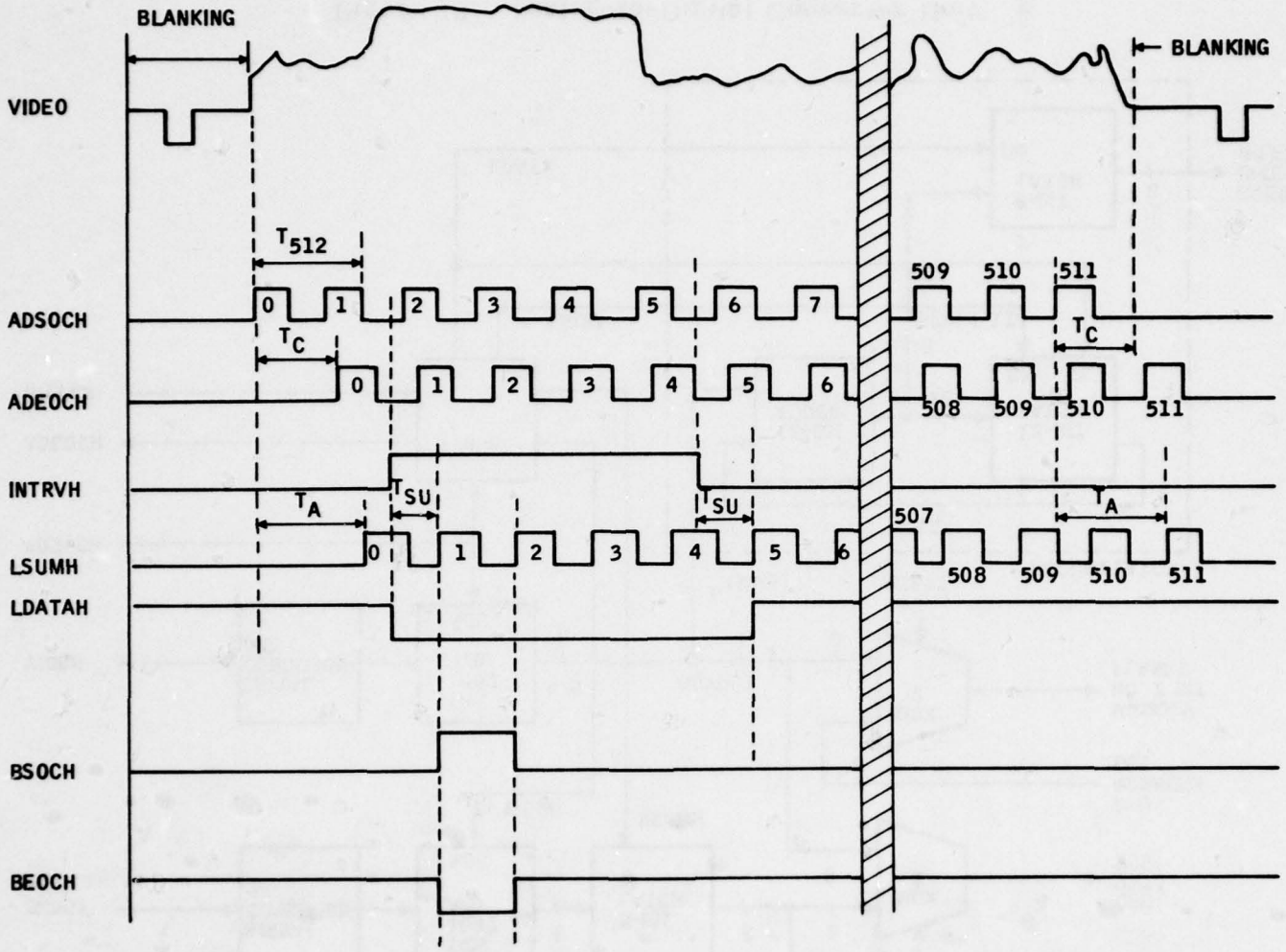


Figure 13. Analog-to-Digital Converter Unit Timing

occurs during the active video and there are exactly 512 clock pulses (samples) per horizontal video line. An end-of-conversion pulse (ADEOCH) is issued back to Memory No. 2 a time after ADSOCH (t_c < 45 nsec).

The latch on the output of the adder is loaded by the LSUMH clock after the data from the A/D has propagated through the adder and set-up on the latch ($t_a \simeq t_c$ + 25 nsec). When INTRVH goes high, a potential interval is enabled. The INTRVH signal is synchronized to the LSUMH clock and a transition period of INTRVH must set up a time greater than the adder propagation delay (t_{su} > 25 nsec). When INTRVH is low, LDATA is high and the 12-bit latch on the add loop is cleared. When INTRVH goes low-to-high, LDATA is set low and the loop can accumulate on following LSUMH clock pulses. When INTRVH goes low, LDATA is set high on the next LSUMH clock pulse, thus latching bits 4-11 of the adder data in the 8-bit latch giving the intensity sum FLF.

The background estimate (BGEST) is digitized at the beginning of every interval. The start-of-conversion pulse (BSOCH) and end-of-conversion pulse (BSOCH) timing is shown in Figure 13. The BSOCH signal goes high at the first low-to-high transition of LSUMH after INTRVH goes high. The duration of BSOCH is one period of the LSUMH clock. When BSOCH goes low, BEOCH goes high and clocks the digitized data into the 8-bit latch on the output of the A/D and the data remains there until the next high-to-low transition of INTRVH.

DIGITAL PROCESSOR

The digital portion of the PATS system is illustrated in Figure 14. All data transfers between subsystems are made through a common, asynchronous data bus. CPU 1 and the two DMA controllers act as bus masters. They initiate bus requests, control bus transfers after the bus is allocated and granted, and then release the bus for possible use by another master. The

two memories and the registers in the DMA controllers act as bus slave units. They cannot initiate any bus action but must respond whenever the appropriate address bits appear on the address lines during a bus cycle.

Whenever a bus master such as the DMA controller wishes to transfer data to a bus slave unit such as Memory No. 1, it must first request use of the bus by signaling the bus arbiter. This circuit checks to see if the bus is in use, and if not, if requests permission to gain control of the bus. If no other bus master is requesting the bus, or if it has highest priority of all requesters, it will be granted the bus and will be allowed to use it as long as necessary to transfer data. When all the data is transferred, the DMA controller signals the arbiter that it is done and the arbiter in turn releases the bus for use by another master. The handshake circuits provide for reliable asynchronous transfer of data between a bus master and a slave unit such as Memory No. 1. The waveforms of Figure 15 illustrate the signal changes that occur during the transfer of one word of data. When data is ready to be transferred and all of the data and address lines have stabilized, the Data Valid line is asserted low. Memory No. 1 can use this signal to write the data into memory and when it has completed this data transfer, it will assert Data Ready low, letting the DMA controller know the transfer has been effected. It, in turn, will set Data Valid high and Memory No. 1 will respond by setting Data Ready high. This double interlocking of control signals provides for increased reliability of data transfer since each unit, master and slave, will give an acknowledgement only when it receives an indication that the other unit has completed its operation. It also provides for the use of memories with different operating speeds since the transfers are asynchronous and a new transfer will not be completed until the necessary acknowledgement signals have been received.

Feature parameters from the interval generation circuits are buffered by the input FIFOs and loaded into Memory No. 1 via the DMA controller. At the end of each line the DMA controller interrupts CPU No. 1 such that it can

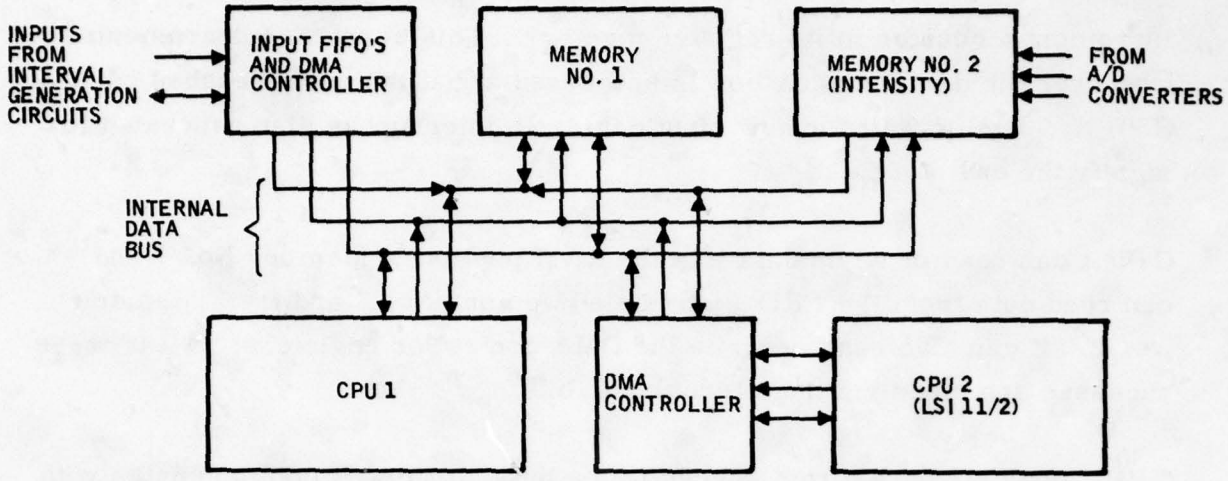


Figure 14. PATS Digital Processing System

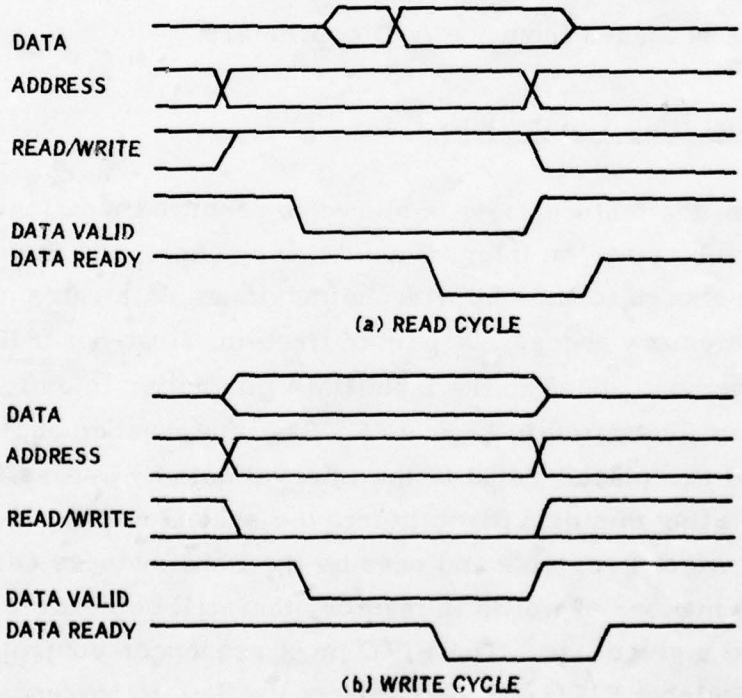


Figure 15. Memory Bus Waveform

increment a counter in its register memory. This counter is decremented whenever the data for each line is processed and if the count reaches zero, CPU No. 1 must wait for new input data. An interrupt is also generated to signify the end of a field.

CPU 1 can read or write data into the RAM portion of Memory No. 1 and can read data from the PROM portion of Memory No. 1 and from Memory No. 2. It can also read or write the DMA controller registers. All of these accesses are made via the internal data bus.

CPU 2 does not have direct access to the bus. Instead it communicates with the remaining portion of the system via a second DMA controller. During the initial phases of system test, both memories will be loaded with precalculated values from CPU 2. Target descriptor outputs from CPU 1 are stored in Memory No. 1 and read via DMA by CPU 2. In normal system operation, Memory No. 2 is loaded from the A/D converters.

Input FIFOs and DMA Controller

Input data from the feature logic is stored in reserved locations in Memory No. 1. However, since an interval can be very short, the memory cannot be loaded fast enough to keep up with the maximum data rates even with the use of direct memory access. A pair of first-in, first-out (FIFO) buffer memories have been added to the input DMA controller to buffer this transfer of input data as illustrated in Figure 16. The line number and interval count for an interval are placed ahead of the interval data by bypassing the first FIFO and inserting this data directly into the second FIFO. Interval count is stored in a second register and used by the DMA address generator to determine the number of words in memory that will be reserved for the interval data for a given line. The FIFO input sequencer controls the shifting of data into the input FIFO and also selects the data to be read-in by selectively controlling the tri-state enables.

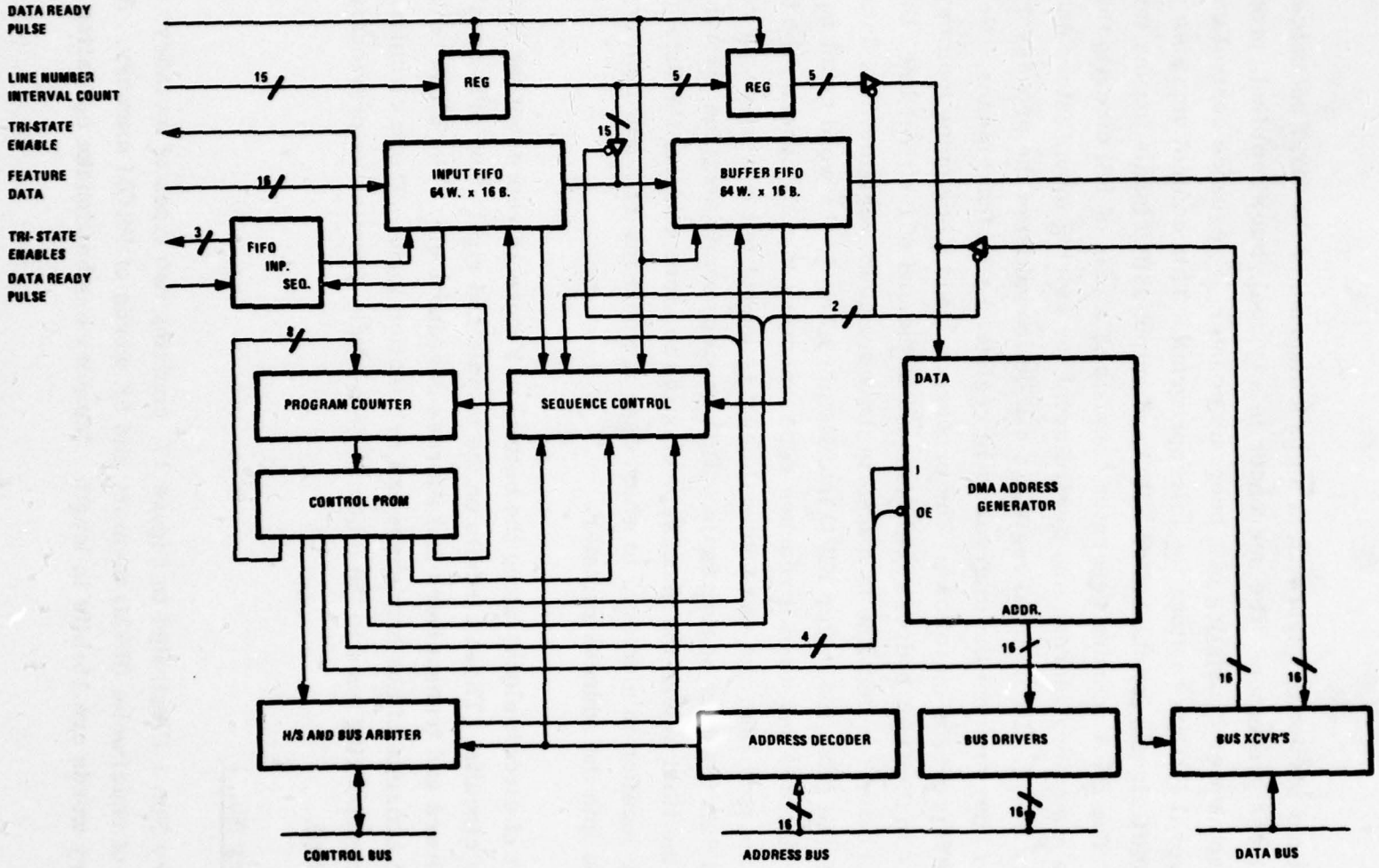


Figure 16. PATS Input FIFOs and DMA Controller

The bottom portion of Figure 16 is a block diagram of the DMA sequencer and address generator. The sequencer is a typical, but simplified, programmable sequencer consisting of a program counter, a sequence control unit, and several PROMs to store the microprogram. This control unit also provides shift signals for the buffer FIFO and output shift signals for the input FIFO. The DMA address generator consists of a pair of LSI circuits that provide a memory address and a word count for storing input feature data in Memory No. 1. Its internal registers can be loaded from the processor via the data bus, or from the interval count register by internal means. Normal operation proceeds as follows: The processor loads a beginning memory address in the DMA address register, and at the end of the first line, the interval count for that line is loaded in the word count register. As data is transferred from the buffer FIFO into Memory No. 1, the word count register is decremented and when it reaches zero, memory loading ceases. At the end of the next line, the new interval count is loaded in the word count register and data transfer begins again. This process would continue until the end of the field unless the processor chooses to store some of the data at another location in memory, in which case it would insert a new memory address into the address register.

The set of circuit blocks along the bottom of Figure 16 represent the bus interface circuits. Those circuits on the middle and right-hand side consist of bus drivers and transceivers and address decoder logic. The block on the far left consists of bus handshake and arbitration logic. These circuits, and the corresponding ones in the other system block, function as previously described.

Memory No. 1

Memory No. 1 illustrated in Figure 17, contains two types of memory: 16K words of read/write (RAM) memory and 8K words of PROM memory. All memory words are 16-bits in length. Memory bus handshake operation is

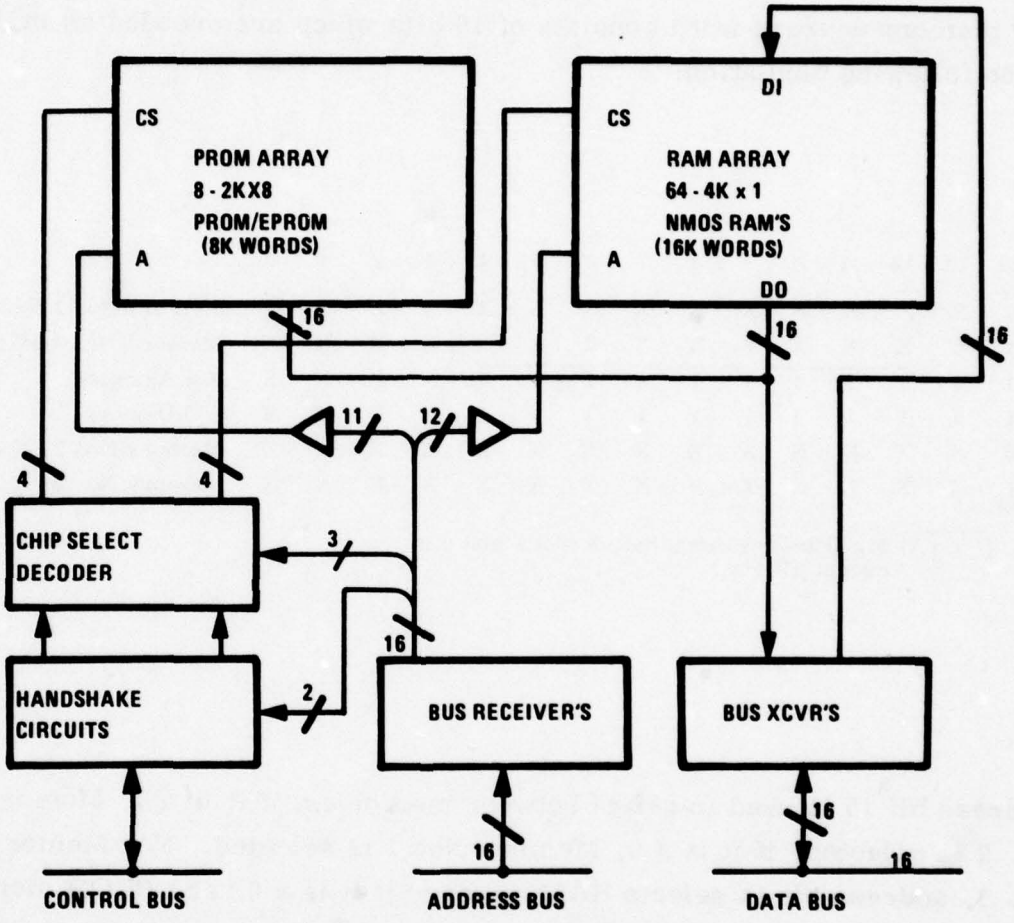


Figure 17. PATS Memory No. 1

as described in a previous subsection. The remaining circuits consist of bus receivers and transceivers, address decoders, and memory drive circuits.

The memory address word consists of 16 bits which are encoded as indicated in the following tabulation:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Memory No. 1 RAM
0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	Memory No. 1 PROM
0	1	1	$\overline{(1}$	1	1	1	1	1	1	1	$\overline{1)}$	X	X	X	X	Not Assigned
0	1	1	1	1	1	1	1	1	1	1	1	X	X	X	X	I/O Devices
1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Memory No. 2, X Address
1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Memory No. 2, Y Address

Note: $\overline{(1\ 1\ 1\ 1)}$ Signifies any combination of 1's and 0's except all 1's.

Address bit 15 is used to select between memories; if it is a 1, Memory No. 2 is selected, if it is a 0, Memory No. 1 is selected. For Memory No. 1, address bit 14 selects RAM memory if it is a 0, and PROM memory if it is a 1. The least significant 14 bits provide addresses for up to 16K words of memory of each type, even though only 8K of PROM memory is requird at the present time. A few of the addresses in the PROM memory space are reserved for I/O devices such as the DMA controller registers.

Processor Control Unit

The processor control unit, illustrated in Figure 18, consists of a clock oscillator and clock control circuit, a microprogram memory and associated sequencer logic, and interrupt logic. A clock oscillator produces a 25-MHz square wave which is then counted down to provide clocks for the processor, memory, DMA circuits, and bus handshake logic. Control inputs permit selection of a single step clock for debugging purposes or a continuous clock for operational use.

An advanced state-of-the-art LSI sequencer chip (AM2910) is employed for control of microinstruction sequencing. Its Y-output is 12 bits wide, thereby enabling it to address up to 4,096 words of microcode, much more than is expected to be needed for this application. All internal elements are a full 12 bits wide. These include the microprogram counter and associated incrementer, a 5-word last in-first out (LIFO) stack for storing subroutine or loop return addresses, and an internal loop counter. The 4-bit instruction code which is normally obtained from the microprogram memory, can be modified by the interrupt logic to permit vectoring to the interrupt service routine. A condition code input facilitiates conditional branching based on the result of a test of up to 16 input parameters. A pipeline register at the output of the microprogram memory allows a new microinstruction to be fetched while the previous one is being executed, thereby substantially increasing the speed of operation. The D-input to the sequencer can either be multiplexed to the Y-output to provide a jump address for the next microinstruction or can be loaded into the loop counter to determine the length of the loop. A multiplexer at the D-input permits selection of data from one of three sources: the microprogram register, a count value from the arithmetic section, or an interrupt vector address. The interrupt logic, in addition to providing these vector addresses, arbitrates between multiple interrupts, selecting the one with the highest priority.

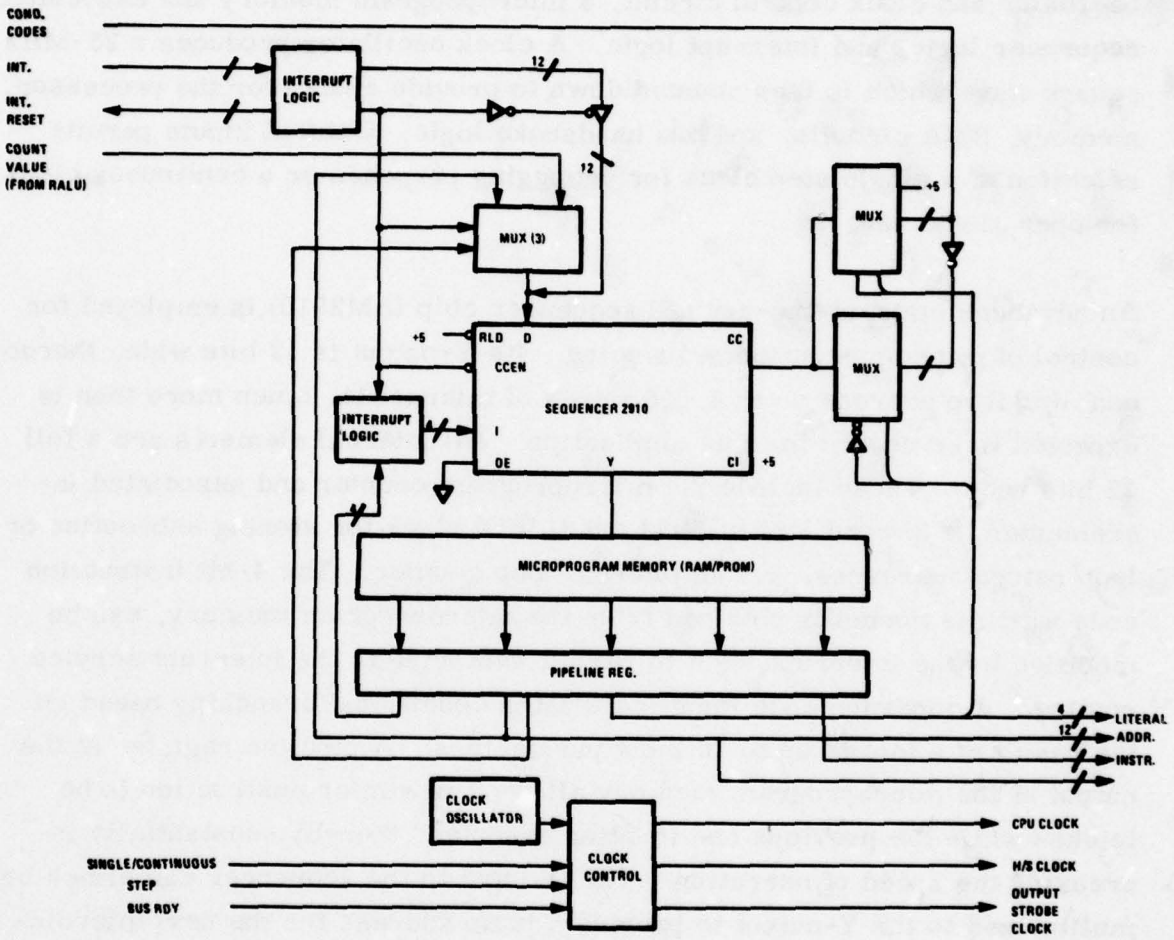


Figure 18. PATS Processor - Control Unit

Register and Arithmetic Logic Unit

The heart of CPU 1 is the register and arithmetic logic unit (RALU) illustrated in Figure 19. A unique feature of this RALU is that it contains two complete 16-bit processing elements, one for performing arithmetic or logical operations on data, and the other for simultaneously calculating the address of the next operand. A third special-purpose processor is included for performing high-speed multiplications and sums of products. Contemporary LSI circuits make parallel processing of this type not only possible but cost effective and practical. Utilization of other LSI and MSI circuits throughout the RALU keeps overall component count to a relatively low value.

Most of the internal inputs come from the microinstruction memory register. They include a 16-bit literal field, a 16-bit address field for the two 16-word register files in the address and operand processor, and a 31-bit instruction field which provides instructions to all sections of the RALU. Outputs to the microinstruction sequencer include a 12-bit count value for the loop counter and a 6-bit status word for use in controlling microinstruction sequencing. The memory bus provides the external interface to the RALU and its operation is identical to that previously described.

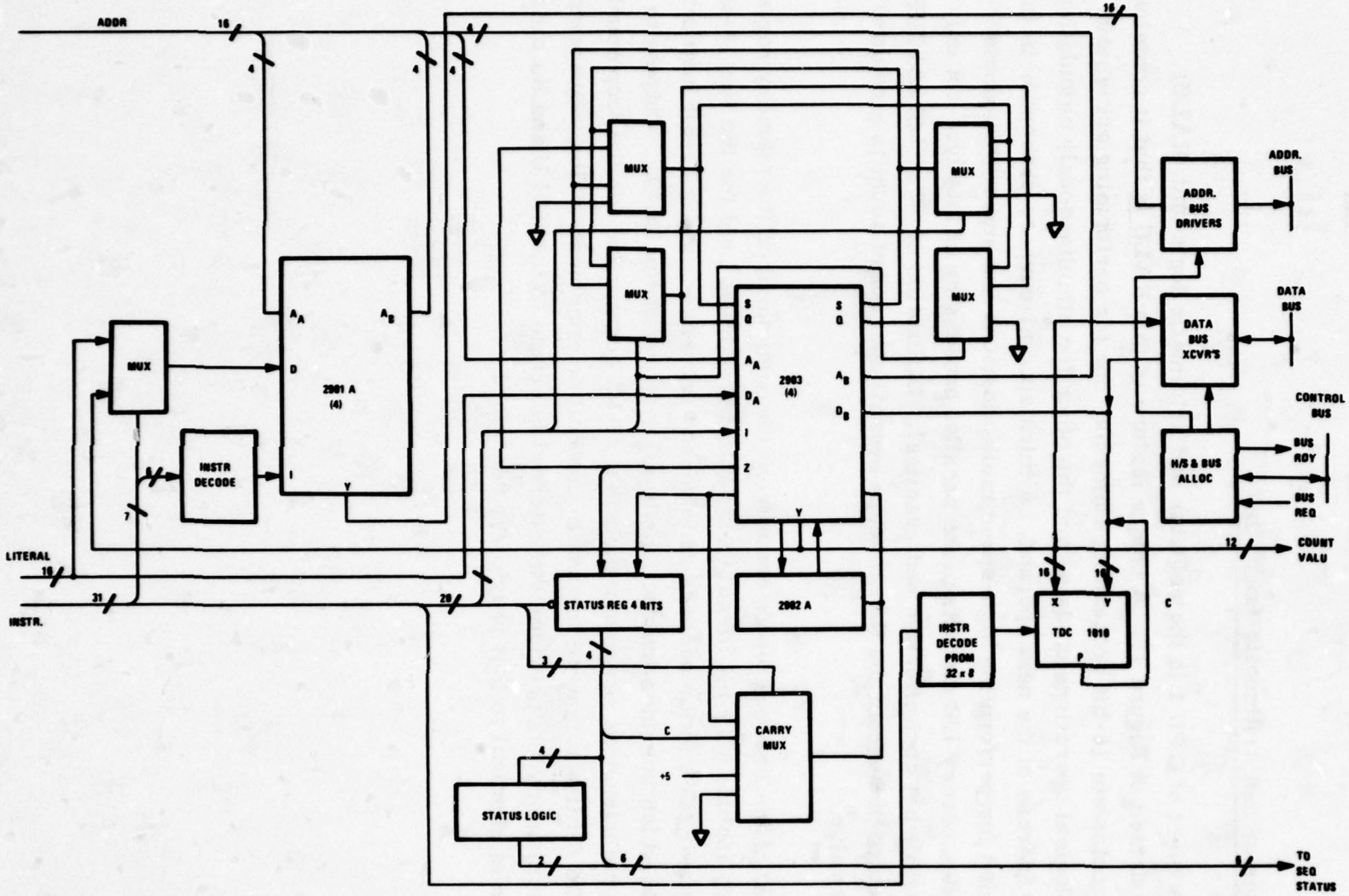


Figure 19. PATS Processor - RALU

SECTION IV

SOFTWARE DESIGN AND CODING

The various modules of the PATS CPU 2 software which were described in the previous report[5] are now being microcoded. A 66-bit microinstruction has been defined to control the architecture described in Section III. The format of the microinstruction will be presented below in detail, together with microcode size estimates and an example piece of code.

MICROINSTRUCTION FORMAT

The model of the microinstruction currently being used to generate microcode is shown in Figure 20. The bits shown may be permuted in the hardware but their definitions will remain the same. The microinstruction is 66 bits long and consists of five major fields, each of which controls a processing function. The fields are as follows:

1. Microsequencer control
2. Memory address generation
3. Data processing
4. Multiplier/accumulator control
5. Memory control.

---

[5] D. E. Soland, et al., PATS Quarterly Progress Report, Contract No. DAAK70-72-C-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, September 1, 1978.

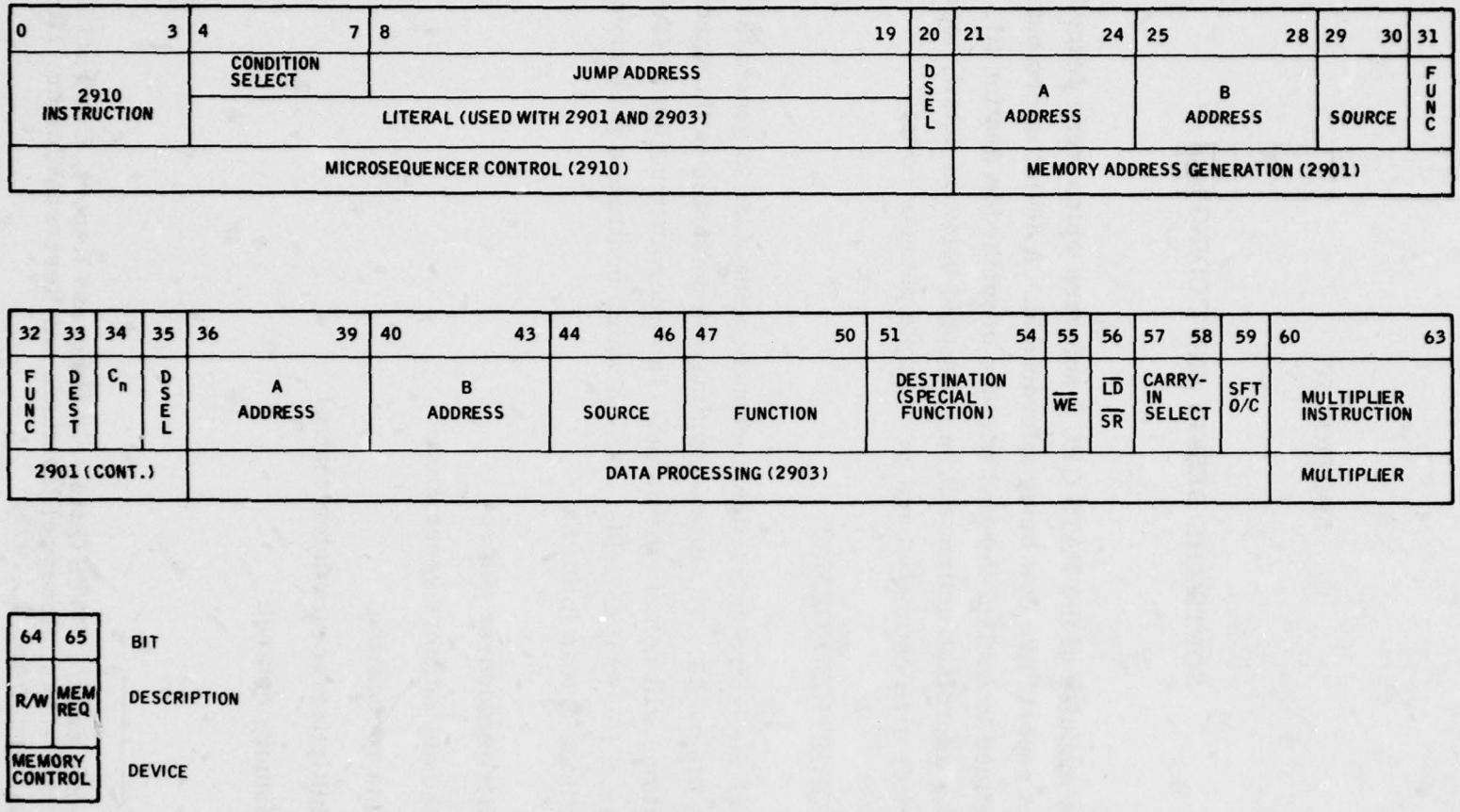


Figure 20. CPU 1 Microinstruction Format (Programming Model)

Each of these fields, in turn, consists of several subfields. Mnemonics have been assigned to the various bit configurations these subfields can take. These mnemonics are used in conjunction with a microassembler which recognizes them and produces the binary output for each microinstruction. In addition, the microassembler allows us to assign a default value to a subfield which is assumed when no data is specified for that subfield. The fields, subfields, mnemonics, and defaults are described below.

Field 1: Microsequencer Control (bits 0-20)

This field controls the sequencing operation through the microinstruction memory. The device being used for this purpose is the AM2910 microprogram controller from Advanced Micro Devices which has a repertoire of 16 instructions.

Subfields:

2910 Instruction (bits 0-3) -- These bits are the instruction (I) input on the 2910 and principally determine the device function (e.g., conditional jumps, conditional jump to subroutine, etc.). The mnemonics for this subfield are shown in the first half of Table 2. A word is in order about the format of this table. The data is read as shown by the microassembler and used to construct a symbol table. The values equated to a symbol, in general, can be either hexadecimal (H#), binary (B#), octal (Q#), or decimal (D#). The comments after the value give the meaning of the mnemonic.

Default: CONT (Continue - no branch)

Branch Condition Select (bits 4-7) -- This subfield selects the condition to be tested for use with a conditional jump. The conditions are shown in Table 2, and refer to the results produced by the AM2903 data processor.

Default: None

TABLE 2. 2910 MNEMONICS

```
;
; 2910 SEQUENCER INSTRUCTIONS
;
JZ:         EQU H#0   ; JUMP ZERO (RESET)
CJS:        EQU H#1   ; CONDITIONAL JUMP SUBROUTINE PIPELINE
JMAP:       EQU H#2   ; JUMP MAP
CJP:        EQU H#3   ; CONDITIONAL JUMP PIPELINE
PUSH:       EQU H#4   ; PUSH/CONDITIONAL LOAD COUNTER
JSRP:       EQU H#5   ; CONDITIONAL JUMP SUBROUTINE R OR PIPELINE
CJV:        EQU H#6   ; CONDITIONAL JUMP VECTOR
JRP:        EQU H#7   ; CONDITIONAL JUMP R OR PIPELINE
RFCT:       EQU H#8   ; REPEAT LOOP, COUNTER NOT ZERO
RPCT:       EQU H#9   ; REPEAT PIPELINE, COUNTER NOT ZERO
CRTN:       EQU H#A   ; CONDITIONAL RETURN
CJPP:       EQU H#B   ; CONDITIONAL JUMP PIPELINE AND POP
LDCT:       EQU H#C   ; LOAD COUNTER AND CONTINUE
LOOP:       EQU H#D   ; TEST END OF LOOP
CONT:       EQU H#E   ; CONTINUE
TWB:        EQU H#F   ; THREE-WAY BRANCH:
                      ; PASS TEST -- CONTINUE PC AND POP STACK
                      ; FAIL TEST -- JUMP PIPELINE AND POP IF COUNTER ZERO
                      ; FAIL TEST -- REPEAT LOOP IF COUNTER NOT ZERO
;
; BRANCH CONDITION CODES
;
ONE:        EQU H#0   ; FORCE BRANCH CONDITION = .TRUE.
ZERO:       EQU H#1   ; BRANCH IF ZERO
NEG:        EQU H#2   ; BRANCH IF NEGATIVE
CARRY:      EQU H#3   ; BRANCH IF CARRY
OVF:        EQU H#4   ; BRANCH IF OVERFLOW
LEZERO:     EQU H#5   ; BRANCH IF LESS THAN OR EQUAL TO ZERO
NONZERO:    EQU H#9   ; BRANCH IF NONZERO
POS:        EQU H#A   ; BRANCH IF POSITIVE
NOCARRY:    EQU H#B   ; BRANCH IF NO CARRY
NOOVF:      EQU H#C   ; BRANCH IF NO OVERFLOW
GTZERO:     EQU H#D   ; BRANCH IF GREATER THAN ZERO
;
; 2910 D INPUT SELECT - SEE 2901 D INPUT SELECT
;
```

Jump Address (bits 8-19) -- This is the address used by a jump instruction or the value to be loaded into the 2910 register/counter by a "load counter" instruction.

Default: None.

D Input Select (bit 20) -- This selects the D (external) input on the 2910 from either the microinstruction (i.e., bits 8-19) or the 2903 Y output. For use with jump instructions, the programmer will normally use this bit to select the jump address from the microinstruction. However, when executing an instruction on the 2910 to load the register/counter, for our purposes it is sometimes necessary to select the output from the 2903 (i.e., we use the register/counter as a loop counter and must initialize it with computed data). The mnemonics for this subfield are shown with the memory address generation (AM2901) mnemonics.

Default: LITERAL (select data from jump address subfield)

When the microsequencer control field is not specified, the defaults cause the sequencer to sequence normally through the microinstructions.

Figure 20 shows the literals subfield spanning bits 4-19. This subfield is used by the 2903 data processor and 2901 memory address generator, but the bits are also shared with the 2910. It is up to the programmer (and to a certain extent, the assembler) to ensure that there are no conflicts in the usage of these bits.

Field 2: Memory Address Generation
(bits 4-19, 21-35)

This field controls the array of AM2901 RALUs for the purpose of generating addresses to the various PATS memories. We do not require all the capabilities of the 2901s for this application and so we're able to encode several of

the subfields necessary to control them. The 2901 allows us to compute an address and store it in a register while simultaneously processing data in the 2903. This "pipelining" effectively removes the overhead of addressing the memories in PATS. See Table 3 for mnemonics.

Subfields:

Literal Data (bits 4-19) -- This subfield is optionally used to supply data for the D (external) input on the 2901. This subfield is used in conjunction with the D-input select subfield (see below).

Default: None

A Address Select (bits 21-24) -- These bits select the register for the A port on the 2901 RAM. The mnemonics for these registers are shown in Table 3.

Default: None

B Address Select (bits 25-28) -- Selects the register for the B port on the 2901 RAM.

Default: None

ALU Operand Sources (bits 29-30) -- These bits select the (R S) source pair for the R and S inputs on the 2901 ALU. We have selected four pairs out of a possible eight.

Default: DZ (select R=D (external input), S=zero)

ALU Function (bits 31-32) -- Selects the ALU function on the R and S inputs selected in the previous subfield. We have selected four out of a possible eight for the device.

TABLE 3. 2901 MNEMONICS

```
;
; 2901 REGISTER DESIGNATORS
;
AR0:        EQU H#0
AR1:        EQU H#1
AR2:        EQU H#2
AR3:        EQU H#3
AR4:        EQU H#4
AR5:        EQU H#5
AR6:        EQU H#6
AR7:        EQU H#7
AR8:        EQU H#8
AR9:        EQU H#9
AR10:       EQU H#A
AR11:       EQU H#B
AR12:       EQU H#C
AR13:       EQU H#D
AR14:       EQU H#E
AR15:       EQU H#F
;
; 2901 SOURCES (RS) -- AB IS DEFINED WITH 2903 SOURCES
;
DZ:         EQU B#01   ; D, ZERO
ZA:         EQU B#10   ; ZERO, RAM A
DA:         EQU B#11   ; D, RAM A
;
; 2901 FUNCTIONS  (R FUNCTION S)
;
ADD.:       EQU B#00   ; R+S+CN
SUBR.:      EQU B#01   ; S-R-1+CN
OR.:        EQU B#11   ; R .OR. S
AND.:       EQU B#10   ; R .AND. S
;
; 2901 DESTINATION CONTROL
;
NOP:        EQU B#0    ; Y=F
RAMF:       EQU B#1    ; Y=F, F INTO RAM B
;
; 2901 CARRY-IN SELECT
; USE CARRY0 AND CARRY1 FROM 2903 CARRYIN SELECT
;
; 2901 D INPUT SELECT/2910 D INPUT SELECT
;
LITERAL:    EQU B#0    ; SELECTS LITERAL (OR JMP ADDR) FROM MICROINSTR
SELECTY:    EQU B#1    ; SELECTS Y OUTPUT FROM 2903
```

Default: AND. (R ∧ S)

ALU Destination (bit 33) -- This bit determines where the ALU result is deposited. For our purposes, this is either the Y output on the 2901 or both the Y output and the register addressed in the B Select subfield.

Default: NOP (output ALU result to Y).

Carry-In (bit 34) -- Specifies the carry-in to the 2901 ALU (CARRY0=0, CARRY1=1).

Default: CARRY0 (carry-in=0)

D-Input Select (bit 35) -- This bit selects the D (external) input on the 2901 from either the microinstruction (literal data-bits 4-19) or the Y output on the 2903 data processor. This is required to implement indirect addressing (i.e., we store address pointers in memory, fetch them with the 2903 and pass them on to the 2901).

Default: LITERAL (select literal data from microinstruction)

When the memory address generation field is not specified, the default conditions cause the 2901 to output zero and hold all internal registers intact. The literal data subfield (bits 4-19) shares bits with the 2910. Conflicts in usage must be avoided.

Field 3: Data Processing (bits 4-19, 36-59)

These bits control the array of AM2903 RALUs for the purpose of processing data fetched from the PATS memories. The 2903 is a second-generation 2901 and offers more functions than the latter device. We are using all of its capabilities, so its control bits take up the lion's share of the microinstruction. Mnemonics are detailed in Table 4.

TABLE 4. 2903 MNEMONICS

```
;
; 2903 REGISTER DESIGNATORS
;
R0:         EQU H#0
R1:         EQU H#1
R2:         EQU H#2
R3:         EQU H#3
R4:         EQU H#4
R5:         EQU H#5
R6:         EQU H#6
R7:         EQU H#7
R8:         EQU H#8
R9:         EQU H#9
R10:        EQU H#A
R11:        EQU H#B
R12:        EQU H#C
R13:        EQU H#D
R14:        EQU H#E
R15:        EQU H#F
;
; 2903 OPERAND SOURCES  (RS)
;
AB:         EQU Q#0   ; RAM A, RAM B, .NOT.OEB=0
ADB:        EQU Q#1   ; RAM A, DB, .NOT.OEB=1
AQ0:        EQU Q#2   ; RAM A, Q, .NOT.OEB=0
AQ1:        EQU Q#3   ; RAM A, Q, .NOT.OEB=1
DAB:        EQU Q#4   ; DA, RAM B, .NOT.OEB=0
DADB:       EQU Q#5   ; DA, DB, .NOT.OEB=1
DAQ0:       EQU Q#6   ; DA, Q, .NOT.OEB=0
DAQ1:       EQU Q#7   ; DA, Q, .NOT.OEB=1
;
; 2903 ALU FUNCTIONS  (R FUNCTION S)
;
SPF:        EQU H#0   ; SPECIAL FUNCTIONS (RS=AB,ADB,DAB,DADB)
HIGH:       EQU H#0   ; FI=HIGH    (RS=AQ0,AQ1,DAQ0,DAQ1)
SUBR:       EQU H#1   ; S-R-1+CN
SUBS:       EQU H#2   ; R-S-1+CN
ADD:        EQU H#3   ; R+S+CN
PASSS:      EQU H#4   ; S+CN
COMPLS:     EQU H#5   ; .NOT.S+CN   (2S COMPLEMENT S)
PASSR:      EQU H#6   ; R+CN
COMPLR:     EQU H#7   ; .NOT.R+CN   (2S COMPLEMENT R)
LOW:        EQU H#8   ; FI=LOW
NOTRS:      EQU H#9   ; .NOT.R.AND.S
EXNOR:      EQU H#A   ; .NOT.(R.EOR.S)
EXOR:       EQU H#B   ; R.EOR.S
AND:        EQU H#C   ; R.AND.S
NOR:        EQU H#D   ; R.NOR.S
NAND:       EQU H#E   ; R.NAND.S
OR:         EQU H#F   ; R.OR.S
```

TABLE 4. 2903 MNEMONICS--CONCLUDED

```
;
; 2903 ALU DESTINATION CONTROL
;
ADR:       EQU H#0   ; ARITH Y=F/2, HOLD Q
LDR:       EQU H#1   ; LOG Y=F/2, HOLD Q
ADRQ:      EQU H#2   ; ARITH Y=F/2, LOG Q=Q/2
LDRQ:      EQU H#3   ; LOG Y=F/2, LOG Q=Q/2
RPT:       EQU H#4   ; Y=F, GENERATE PARITY, HOLD Q
LDQP:      EQU H#5   ; Y=F, LOG Q=Q/2, GENERATE PARITY
QPT:       EQU H#6   ; Y=F, Q=F, GENERATE PARITY  (.NOT.WRITE=H)
RQPT:      EQU H#7   ; Y=F, Q=F, GENERATE PARITY  (.NOT.WRITE=L)
AUR:       EQU H#8   ; ARITH Y=2F, HOLD Q
LUR:       EQU H#9   ; LOG Y=2F, HOLD Q
AURQ:      EQU H#A   ; ARITH Y=2F, LOG Q=2Q
LURQ:      EQU H#B   ; LOG Y=2F, LOG Q=2Q
YBUS:      EQU H#C   ; Y=F, HOLD Q
LUQ:       EQU H#D   ; Y=F, LOG Q=2Q
SINSX:     EQU H#E   ; SIGN EXTEND
REG:       EQU H#F   ; RESULTS TO RAM, SIGN EXTEND
;
; 2903 SPECIAL FUNCTIONS
;
USMUL:     EQU H#0   ; UNSIGNED MULTIPLY
TCMUL:     EQU H#2   ; TWO'S COMPLEMENT MULTIPLY
INCTWO:    EQU H#4   ; INCREMENT BY ONE OR TWO
SMTC:      EQU H#5   ; SIGN MAGNITUDE -- TWO'S COMPLEMENT
TCMLS:     EQU H#6   ; TWO'S COMPLEMENT MULTIPLY, LAST CYCLE
SLN:       EQU H#8   ; SINGLE LENGTH NORMALIZE
DLN:       EQU H#A   ; DOUBLE LENGTH NORMALIZE AND FIRST DIVIDE OP
TCDIV:     EQU H#C   ; TWO'S COMPLEMENT DIVIDE
TCDC:      EQU H#E   ; TWO'S COMPLEMENT DIVISION CORRECTION
;
; 2903 SHIFT CONTROL
;
OPEN:      EQU B#0   ; OPEN SHIFT
CLOSED:    EQU B#1   ; CLOSED (CIRCULAR) SHIFT
;
; 2903 RAM WRITE ENABLE
;
WRITEB:    EQU B#0   ; WRITE INTO RAM B
NOWRITEB:  EQU B#1   ; NO WRITE INTO RAM
;
; STATUS REGISTER LOAD CONTROL
;
ENSTAT:    EQU B#0   ; ENABLE LOAD OF STATUS REGISTER
DISSTAT:   EQU B#1   ; DISABLE LOAD
;
; 2903 CARRY IN SELECT
;
CARRY0:    EQU B#00  ; CARRY IN = 0
CARRY1:    EQU B#01  ; CARRY IN = 1
CZERO:     EQU B#10  ; CARRY IN = Z FLAG FROM 2903
CSTAT:     EQU B#11  ; CARRY IN = C FLAG FROM STATUS REGISTER
;
```

Subfields:

Literal Data (bits 4-19) -- This subfield is optionally used to supply data for the D (external) input on the 2903. This subfield is shared with the 2901.

Default: None

A Address Select (bits 36-39) -- Selects the register for the A port on the 2903 RAM. The mnemonics for these registers are shown in Table 4.

Default: None

B Address Select (bits 40-43) -- Selects the register for the B port on the 2903 RAM.

Default: None

ALU Operand Sources (bits 44-46) -- Selects the (R S) source pair for the R and S inputs on the 2903 ALU.

Default: ADB (R = RAM A, S = DB)

ALU Function (bits 47-50) -- Selects the ALU function on the R and S inputs selected in the previous subfield. Also, this subfield is used in conjunction with the source select subfield to indicate when special ALU functions are to be performed.

Default: LOW (set ALU output to zero)

ALU Destination or Special Function (bits 51-54) -- When the ALU function subfield is zero and certain ALU sources are selected (see Table 4), this subfield specifies the special function to be performed by the ALU. Otherwise, it simply controls ALU destination. (Note: In Table 4 under "ALU Destination," F is the (unshifted) output from the ALU. Principally,

this subfield controls the shift function on the ALU output, and is used in conjunction with subfield 10 below).

Default: YBUS (put F unshifted on to the Y output)

Register Write Enable (bit 55) -- When this bit is 0, data on the Y output is written into the register specified as the B address in subfield 3.

Default: NOWRITEB (disable write into register file)

Status Register Load Enable (bit 56) -- When this bit is 0, the status register is loaded with the current status bits from the 2903 ALU. The bits from this status register are used to form the branch condition selected by subfield 2 of the microsequencer control field.

Default: ENSTAT (enable load of status register)

Carry-In Select (bits 57-58) -- Selects the carry-in to the 2903 ALU.

Default: CARRY0 (carry-in = 0)

Shift Control (bit 59) -- This bit specifies whether a shift performed by the 2903 ALU shifter is to be open or closed (circular). This bit is used in conjunction with ALU Destination Control (subfield 6) to perform left/right, logical/arithmetic, single word/double word, open/closed shifts. Logic external to the 2903 decodes the destination subfield to determine whether a shift is left/right and whether it involves a single register or a RAM register - Q register combination, and sets up inputs to the 2903 needed to accomplish the selected shift. Arithmetic and logical single register shifts are performed directly by the ALU shifter and require no external logic.

Default: OPEN (open shift)

When this field is not selected, defaults leave all internal 2903 registers intact, and the Y output is zero. Note again that the literals subfield (bits 4-19) shares bits with the 2910 and the 2901, so conflicts in usage must be avoided.

Field 4: Multiplier/Accumulator Control
(bits 60-63)

This field controls the TRW TDC1010 multiplier/accumulator. We have devised an instruction set for controlling this device which will be decoded in PROM. The mnemonics for these instructions are shown in Table 5.

Default: NOPMPR (no operation)

Field 5: Memory Control (bits 64-65)

These bits request memory and specify whether a read from or a write to memory is being performed by the 2903. Bits for selecting and controlling a particular memory in PATS (e.g., Memory 1, Memory 2, etc.) are on the high order bits of the memory address bus. The programmer uses the 2901 to put these control bits on the bus together with the memory address. Instruction mnemonics are shown in Table 6.

Default: NOMEM (memory not requested)

MICROCODE EXAMPLE

The median filter has been coded in microassembly language and the microassembler output is shown in Appendix A. The assembler used is the AMDASM assembler from Advanced Micro Devices. Each microinstruction is constructed by defining and overlaying the individual microinstruction fields discussed. Each field is defined as a full-width microinstruction,

TABLE 5. MULTIPLIER/ACCUMULATOR MNEMONICS

```
;
; MULTIPLIER INSTRUCTIONS
;
NOPMPR:    EQU H#0   ; NO OPERATION
PRELOAD:   EQU H#2   ; PRELOAD DATA
LDXY:      EQU H#3   ; LOAD X AND Y, NO MULTIPLY
MPY:       EQU H#4   ; HOLD X AND Y, MULTIPLY
LDYMPY:    EQU H#5   ; LOAD Y, MULTIPLY
LDXMPY:    EQU H#6   ; LOAD X, MULTIPLY
LDXYMPY:   EQU H#7   ; LOAD X AND Y, MULTIPLY
LSP:       EQU H#8   ; OUTPUT LEAST SIGNIFICANT PRODUCT (LSP)
MSP:       EQU H#9   ; OUTPUT MOST SIGNIFICANT PRODUCT (MSP)
COMPL:     EQU H#A   ; ENABLE TWO'S COMPLEMENT
ENACC:     EQU H#B   ; ENABLE ACCUMULATE
LSPLDY:    EQU H#C   ; OUTPUT LSP AND LOAD Y
MSPLDY:    EQU H#D   ; OUTPUT MSP AND LOAD Y
UNSGNED:   EQU H#E   ; DISABLE TWO'S COMPLEMENT (UNSIGNED)
NOACC:     EQU H#F   ; DISABLE ACCUMULATE
;
```

TABLE 6. MEMORY CONTROL MNEMONICS

```
; MEMORY CONTROL
;
WRITE:     EQU B#01  ; REQUEST AND WRITE MEMORY
READ:      EQU B#11  ; REQUEST AND READ MEMORY
NOMEM:     EQU B#00  ; MEMORY NOT REQUESTED
```

generally with those bits unrelated to the field being defined being set to "don't cares" (X's). Table 7 shows all the field definitions we have currently developed. In addition to the "don't cares," bits may also be defined as constants or as variable subfields (indicated by Vs) whose values are defined at assembly time. A variable subfield may also be assigned a default value which is assumed when no substitution is made for that subfield. Each DEF is given a name (e.g., AM2910). Each subfield within a DEF has a definite length which is usually indicated by a leading number in the subfield descriptor (e.g., 4X = 4 bits, 1VB #0 = 1 bit, etc.). The bit positions within a DEF correspond exactly with those in Figure 20.

The clutter feature computations and clutter classifier have been coded in a register transfer notation which translates directly into microassembly language. This translation effort is currently underway.

MICROCODE SIZE ESTIMATE

Current microcode size estimates for the various modules in the field-operational version of PATS are as follows:

Module	Estimated Size in Microinstructions
Bin/interval matching	350
Clutter features/clutter classifier	100
Median filter	85
Moment features	250
Recognition classifier	350
	1,135

It is currently expected that the final version of the firmware will fit in less than 2K of 72-bit control store.

TABLE 7. MICROASSEMBLER FORMAT DEFINITIONS

```
;
; DEFINITIONS OF OVERLAY FIELDS
;
AM2910:    DEF 4VH#E,4VX,12V%X,1VB#0,45X
AM2901:    DEF 4X,16V%X,1X,4VX,4VX,2V%:B#01,2VB#10,1VB#0,1V%:B#0,
/              1VB#0,30X
MEM:       DEF 64X,2VB#00
MPR:       DEF 60X,4VH#0,2X
AM2903:    DEF 4X,16V%X,16X,4VX,4VX,3VQ#1,4VH#8,4VH#C,1VB#1,
/              1VB#1,2VB#00,1VX,6X
;
;
; SPECIAL FORMATS - 2903
;
; RA F RB -> RB  (RFRR RA,RB,F [,STATUS,CARRYIN])
RFRR:      DEF 36X,4VX,4VX,Q#0,4VX,H#C,B#0,1VB#0,2VB#00,7X
;
; RA F RB -> Y  (RFRY RA,RB,F [,STATUS,CARRYIN])
RFRY:      DEF 36X,4VX,4VX,Q#0,4VX,H#C,B#1,1VB#0,2VB#00,7X
;
; LITERAL F RB -> RB  (LFRR LITERAL,RB,F [,STATUS,CARRYIN])
LFRR:      DEF 4X,16V%X,20X,4VX,Q#4,4VX,H#C,B#0,1VB#0,2VB#00,7X
;
; LITERAL F RB -> Y  (LFRY LITERAL,RB,F [,STATUS,CARRYIN])
LFRY:      DEF 4X,16V%X,20X,4VX,Q#4,4VX,H#C,B#1,1VB#0,2VB#00,7X
;
; SPECIAL FORMATS - 2901
;
; INCREMENT AR  (INCAR AR,AR)
INCAR:     DEF 21X,4VX,4VX,B#10,B#00,B#1,B#1,31X
;
; ARA + ARB -> ARB  (ADDAR ARA,ARB)
ADDAR:     DEF 21X,4VX,4VX,B#00,B#00,B#1,B#0,31X
;
; LITERAL + ARA -> ARB  (ADDLAR LITERAL,ARA,ARB)
ADDLAR:    DEF 4X,16V%X,1X,4VX,4VX,B#11,B#00,B#1,B#0,B#0,30X
;
; LITERAL -> AR  (MOVLAR LITERAL,AR)
MOVLAR:    DEF 4X,16V%X,5X,4VX,B#01,B#11,B#1,1X,B#0,30X
;
; LITERAL + AR -> Y  (ADDLARY LITERAL,AR)
ADDLARY:   DEF 4X,16V%X,1X,4VX,4X,B#11,B#00,B#0,B#0,B#0,30X
;
; AR -> Y  (MOVARY AR)
MOVARY:    DEF 21X,4VX,4X,B#10,B#11,B#0,32X
```

SECTION V

PLANS FOR THE NEXT REPORTING PERIOD

During the next reporting period, all circuit boards for the target screener portion of the system will be completed and checkout will begin. The image-enhancement circuit boards and the symbol-generation board will be built later since they are not needed for the integration and checkout of the target screener hardware and software. Also, microprogram coding and code checkout will continue.

REFERENCES

1. D. E. Soland, et al., "Quarterly Progress Report, Prototype Automatic Target Screener," Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, Minn., January 15 1978.

2. Ibid, p. 23.

3. Ibid, pages 45-47.

4. D. E. Soland et al., "Quarterly Progress Report, Prototype Automatic Target Screener," Contract No. DAAK70-77-C-0248, Honeywell Systems and Research Center, Minneapolis, Minn., 15 June 1978.

5. D. E. Soland, et al., PATS Quarterly Progress Report, Contract No. DAAK70-72-C-0248, Honeywell Systems and Research Center, Minneapolis, Minnesota, September 1, 1978.

APPENDIX A

PATS MEDIAN FILTER

```
AMD AMDASM MICRO ASSEMBLER, V2.0                                PAGE  1
PATS MEDIAN FILTER              9/25/78
      ;
      ; MEDIAN FILTER
      ;
      ; INPUT:
      ;   BIN DATA BLOCK
      ;   AR0 = BIN ORIGIN
      ;   R11 = TOTAL LENGTH OF BIN
      ; OUTPUT:
      ;   DATA BLOCK STARTING AT 2000H CONTAINING INTERLEAVED
      ;   MEDIAN FILTERED STARTING X'S AND WIDTHS FOR THE
      ;   INTERVALS IN THE BIN.
      ;
      ; INITIALIZE OUTPUT ADDR TO 2000H
 0000       AM2901 H#2000,,AR2,DZ,OR.,RAMF ;AR2<-H#2000
      /   & AM2903 & AM2910 & MPR & MEM
      ;
      ; TOTAL LENGTH : 3
 0001       AM2903 D#3,,R11,DAB,SUBR,YBUS,,ENSTAT,CARRY1 ;2903Y<-R11-3
      /   & AM2901 & AM2910 & MPR & MEM
      ;
      ; READ FIRST STARTING X VALUE
 0002       AM2901 D#9,AR0,AR1,DA,ADD.,RAMF & MEM READ ;AR1<-AR0+9,READ
      /   & AM2903 & AM2910 & MPR
      ;
      ; EXIT IF LENGTH < 3
 0003       AM2910 CJP,NEG,EXIT & AM2903 & AM2901 & MPR & MEM
      ;
      ; OBJECT IS BIG ENOUGH TO BE FILTERED
      ; READ IN AND SORT FIRST 3 X VALUES IN R0,R1,R2
 0004       AM2910 CJS,ONE,SORT3 ;JMP TO SUBR SORT3
      /   & AM2901 & AM2903 & MPR & MEM
      ;
      ; READ IN FIRST VALUE OF X+WIDTH AND STORE MIDDLE VALUE FROM SORT
IN R10
      ; AND READ IN AND SORT FIRST 3 VALUES OF X+WIDTH USING R0,R1,R2
 0005       AM2901 ,AR1,AR1,ZA,ADD.,RAMF,CARRY1 & MEM READ ;AR1<-AR1+1,R
EAD
      /   & AM2903 ,R1,R10,AB,PASSR,YBUS,WRITEB ;R10<-R1
      /   & AM2910 CJS,ONE,SORT3 & MPR ;JMP TO SUBR SORT3
      ;
      ; WRITE FIRST TWO SETS OF FILTERED EDGE DATA IN MEM
 0006       AM2901 ,AR2,,ZA,OR.,NOP & MEM WRITE ;2901Y<-AR2,WRITE
      /   & AM2903 ,R10,,AB,PASSR,YBUS & AM2910 & MPR ;2903Y<-R10

 0007       AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 & MEM WRITE ;AR2<-AR2+1,
WRITE
      /   & AM2903 ,R1,R10,AB,SUBS,YBUS & AM2910 & MPR;2903Y<-R1-R10

 0008       AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 & MEM WRITE ;AR2<-AR2+1,
WRITE
      /   & AM2903 ,R10,,AB,PASSR,YBUS & AM2910 & MPR ;2903Y<-R10

 0009       AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 & MEM WRITE ;AR2<-AR2+1,
WRITE
      /   & AM2903 ,R1,R10,AB,SUBS,YBUS & AM2910 & MPR;2903Y<-R1-R10
      ;
      ; CHECK FOR LENGTH = 3
 000A       AM2903 D#3,,R11,DAB,SUBR,YBUS,,ENSTAT,CARRY1 ;2903Y<-R11-3
      /   & AM2901 & AM2910 & MPR & MEM
 000B      AM2910 CJP,GTZERO,A & AM2901 & AM2903 & MPR & MEM ;IF R11>3
GOTO A
```

```
      ;
      ; LENGTH = 3 - WRITE OUT 3RD SET OF FILTERED EDGES AND EXIT
 000C       AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 & MEM WRITE ;AR2<-AR2+1,
WRITE
      /   & AM2903 ,R10,,AB,PASSR,YBUS & AM2910 & MPR ;2903Y<-R10

 000D       AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 & MEM WRITE ;AR2<-AR2+1,
WRITE
      /   & AM2903 ,R1,R10,AB,SUBS,YBUS ;2903Y<-R1-R10
      /   & AM2910 CJP,ONE,EXIT ;GOTO EXIT
      ;
      ; DOES OBJECT HAVE LENGTH = 4?
      ;
 000E A:    AM2901 D#13,AR0,AR1,DA,ADD.,RAMF & MEM READ ;AR1<-AR0+13,REA
D
      /   & AM2910 & AM2903 & MPR

 000F       AM2903 D#4,,R11,DAB,SUBR,YBUS,,ENSTAT ;2903Y<-R11-4
      /   & AM2910 LDCT,,,SELECTY & AM2901 & MPR & MEM ;CNTR<-2903Y

 0010       AM2910 CJP,ZERO,C & AM2901 & AM2903 & MPR & MEM ;IF R11=4 GO
TO C
      ;
      ; OBJECT LENGTH IS >4
      ; OBJECT CAN BE FILTERED USING FILTER SIZE = 5
      ;
      ; READ IN AND SORT 5 X VALUES USING R0 - R4
      ;
 0011 B:    AM2910 CJS,ONE,SORT5 ;JMP TO SUBR SORT5
      ;
      ; STORE MIDDLE VALUE IN TEMP REGISTER
      ;
 0012       AM2903 ,R2,R10,AB,PASSR,YBUS,WRITEB & MPR
      /    & AM2901 ,AR1,AR1,ZA,ADD.,RAMF,CARRY1 & MEM READ & AM2910
      ;      R2 TO R10, AR1+1 TO AR1, READ DATA
      ;
      ; READ IN AND SORT 5 VALUES OF X + WIDTH USING R0-R4,
      ; JUMP TO SUBROUTINE SORT5, STORE RESULTS

 0013       AM2910 CJS,ONE,SORT5 & AM2903 & AM2901 & MEM & MPR
      ;
 0014       AM2903 ,R10,,AB,PASSR,YBUS & AM2901 ,AR2,AR2,ZA,ADD.,RAMF,C
ARRY1
      /    & AM2910 & MEM WRITE & MPR ; R10 TO Y, AR2+1 TO AR2, WRITE DA
TA
      ;
 0015       AM2903 ,R2,R10,AB,SUBS,YBUS,,CARRY1 & MEM WRITE & AM2910
      /    & MPR & AM2901 ,AR2,AR2,ZA,ADD.,RAMF,CARRY1 ; R2-R10 TO Y
      ;      AR2+1 TO AR2, WRITE DATA
      ;
 0016       AM2901 D#4,AR1,AR1,DA,ADD.,RAMF & AM2903 & MEM READ & MPR
      ;      AR1+4 TO AR1, READ DATA
      ;
      ; DECREMENT LOOP COUNTER -- IF NONZERO, GO TO B

      ;      AM2910 RPCT,,B & AM2901 & AM2903 & MEM & MPR
      ;
 0017       AM2901 D#5,AR1,AR1,DA,ADD.,RAMF & AM2903 & MEM READ & AM291
0
      /    & MPR ; AR1+5 TO AR1, READ DATA
      ;
      ; READ IN AND SORT LAST 3 STARTING X VALUES USING R0, R1, R2
```

```
      ;
0018          AM2910 RPCT,,B & AM2903 & AM2901 & MEM & MPR
      ;
      ; TAKE CARE OF LOWER BOUNDARY
      ;
0019          AM2901 D#5,AR1,AR1,DA,ADD.,RAMF & AM2903 & MEM READ & AM291
0
      /     & MPR ; AR1+5 TO AR1, READ DATA
      ;
      ; READ IN AND SORT LAST 3 STARTING X VALUES USING R0, R1, R2
      ; JUMP TO SUBROUTINE SORT3
      ;
001A C:       AM2910 CJS,ONE,SORT3 & AM2903 & AM2901 & MEM & MPR
      ;
      ; STORE MIDDLE VALUE IN TEMP REGISTER
      ;
001B          AM2903 ,R1,R10,AB,PASSR,YBUS,WRITEB & AM2910
      /     & AM2901 ,AR1,AR1,ZA,ADD.,RAMF,CARRY1 & MEM READ & MPR
      ;       R1 TO R10, AR1+1 TO AR1, READ DATA
      ;
      ; READ IN AND SORT THE LAST 3 VALUES OF X + WIDTH USING R0,R1,R2
      ;
001C          AM2910 CJS,ONE,SORT3 & AM2903 & AM2901 & MEM & MPR
      ;
001D          AM2903 ,R10,,AB,PASSR,YBUS & AM2901 ,AR2,AR2,ZA,ADD.,RAMF,C
ARRY1
      /     & MEM WRITE & AM2910 & MPR ; R10 TO Y, AR2+1 TO AR2, WRITE DA
TA
      ;
001E          AM2903 ,R1,R10,AB,SUBS,YBUS,,CARRY1 & AM2901 ,AR2,AR2,ZA,AD
D.,RAMF,
      /       CARRY1 & MEM WRITE & AM2910 & MPR ; R1-R10 TO Y, AR2+1 TO A
R2,
      ;       WRITE DATA
      ;
001F          AM2903 ,R10,,AB,PASSR,YBUS & AM2901 ,AR2,AR2,ZA,ADD.,RAMF,C
ARRY1
      /     & MEM WRITE & AM2910 & MPR ; R10 TO Y, AR2+1 TO AR2, WRITE DA
TA
      ;
0020          AM2903 ,R1,R10,AB,SUBS,YBUS,,CARRY1 & AM2901 ,AR2,AR2,ZA,AD
D.,RAMF,
      /       CARRY1 & MEM WRITE & AM2910 CJS,ONE,EXIT & MPR ; R1-R10 TO
Y,
      ;       AR2+1 TO AR2, WRITE DATA, GO TO EXIT
      ;
0021 SORT3: AM2903 ,R0,,ADB,PASSR,WRITEB & AM2901 D#5,AR1,,DA,ADD.,NOP
      /     & MEM READ & AM2910 & MPR ; DATA TO R0, AR1+5 TO Y, READ DATA
      ;
0022          AM2903 ,R1,,ADB,PASSR,YBUS,WRITEB & AM2901 D#10,AR1,,DA,ADD
.,NOP
      /     & MEM READ & AM2910 & MPR ; DATA TO R1, AR1+10 TO Y, READ DAT
A
      ;
0023          AM2903 ,,R2,ADB,PASSS,YBUS,WRITEB & AM2901 & AM2910 & MEM
      /     & MPR ; DATA TO R2
      ;
0024          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
      ;
0025          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR
      ;       IF RESULT >0, CALL XCH01
```

```
      ;
0026          AM2903 ,R1,R2,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R1-R2 TO Y, ENABLE STATUS
      ;
0027          AM2910 CJS,GTZERO,XCH12 & AM2903 & AM2901 & MEM & MPR
      ;       IF RESULT >0, CALL XCH12
      ;
0028          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
      ;
0029          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR
      ;       IF RESULT >0, CALL XCH01
      ;
002A          AM2910 CRTN,ONE & AM2901 & AM2903 & MEM & MPR ; RETURN FROM
SUBROUTINE
      ;
002B SORT5: AM2903 ,,R0,ADB,PASSS,YBUS,WRITEB & AM2901 D#5,AR1,,DA,ADD.
,NOP
      /     & AM2910 & MEM READ & MPR ; DATA TO R0, AR1+5 TO Y, READ DATA
      ;
002C          AM2903 ,,R1,ADB,PASSS,YBUS,WRITEB & AM2901 D#10,AR1,,DA,ADD
.,NOP
      /     & AM2910 & MEM READ & MPR ; DATA TO R1, AR1+10 TO Y, READ DAT
A
      ;
002D          AM2903 ,,R2,ADB,PASSS,YBUS,WRITEB & AM2901 D#15,AR1,,DA,ADD
.,NOP
      /     & MEM READ & AM2910 & MPR ; DATA TO R2, AR1+15 TO Y, READ DAT
A
      ;
002E          AM2903 ,,R3,ADB,PASSS,YBUS,WRITEB & AM2901 D#20,AR1,,DA,ADD
.,NOP
      /     & MEM READ & AM2910 & MPR ; DATA TO R3, AR1+20 TO Y, READ DAT
A
      ;
002F          AM2903 ,,R4,ADB,PASSS,YBUS,WRITEB & AM2901 & AM2910 & MEM
      /     & MPR ; DATA TO R4
      ;
0030          AM2903 H#FFFF,R10,DAB,PASSS,YBUS,WRITEB & AM2901 & AM2910
      /     & MEM & MPR ; -1 TO R10, ENABLE STATUS
      ;
0031          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
      ;
0032          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR ;
      ;       IF RESULT >0, CALL XCH01
      ;
0033          AM2903 ,R1,R2,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R1-R2 TO Y, ENABLE STATUS
      ;
0034          AM2910 CJS,GTZERO,XCH12 & AM2903 & AM2901 & MEM & MPR
      ;       IF RESULT >0, CALL XCH12
      ;
0035          AM2903 ,R2,R3,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R2-R3 TO Y, ENABLE STATUS
      ;
0036          AM2910 CJS,GTZERO,XCH23 & AM2903 & AM2901 & MEM & MPR
      ;       IF RESULT >0, CALL XCH23
      ;
0037          AM2903 ,R3,R4,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
      /     & MEM & MPR ; R3-R4 TO Y, ENABLE STATUS
      ;
```

```
 0038          AM2910 CJS,GTZERO,XCH34 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH34
       ;
 0039          AM2903 ,,R10,AB,PASSS,YBUS,,ENSTAT & AM2901 & AM2910 & MEM
& MPR
       ;       R10 TO Y, ENABLE STATUS
       ;
 003A          AM2903 ,R10,R10,AB,SUBS,YBUS,WRITEB & AM2901 & MEM & MPR
       /     & AM2910 CRTN,NEG ; -1 TO R10, IF RESULT <0, RETURN
       ;
 003B          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
       ;
 003C          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH01
       ;
 003D          AM2903 ,R1,R2,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R1-R2 TO Y, ENABLE STATUS
       ;
 003E          AM2910 CJS,GTZERO,XCH12 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH12
       ;
 003F          AM2903 ,R2,R3,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R2-R3 TO Y, ENABLE STATUS
       ;
 0040          AM2910 CJS,GTZERO,XCH23 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH23
       ;
 0041          AM2903 ,,R10,AB,PASSS,YBUS,,ENSTAT & AM2901 & AM2910
       /     & MEM & MPR ; R10 TO Y, ENABLE STATUS
       ;
 0042          AM2910 CRTN,NEG & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT <0, RETURN FROM SUBROUTINE
       ;
 0043          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
       ;
 0044          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH01
       ;
 0045          AM2903 ,R1,R2,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R1-R2 TO Y, ENABLE STATUS
       ;
 0046          AM2910 CJS,GTZERO,XCH12 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH12
       ;
 0047          AM2903 ,R0,R1,AB,SUBS,YBUS,,ENSTAT,CARRY1 & AM2901 & AM2910
       /     & MEM & MPR ; R0-R1 TO Y, ENABLE STATUS
       ;
 0048          AM2910 CJS,GTZERO,XCH01 & AM2903 & AM2901 & MEM & MPR
       ;       IF RESULT >0, CALL XCH01
       ;
 0049          AM2910 CRTN,ONE & AM2903 & AM2901 & MEM & MPR
       ;       RETURN FROM SUBROUTINE
       ;
 004A XCH01:   AM2903 ,R0,R10,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
       /     & MEM & MPR ; R0 TO R10
       ;
 004B          AM2903 ,R1,R0,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
       /     & MEM & MPR ; R1 TO R0
       ;
 004C          AM2903 ,R10,R1,AB,PASSR,YBUS,WRITEB & AM2910 CRTN,ONE & AM2
```

```
901
      /    & MEM & MPR ; R10 TO R1, RETURN FROM SUBROUTINE
      ;
 004D XCH12: AM2903 ,R1,R10,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R1 TO R10
      ;
 004E        AM2903 ,R2,R1,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R2 TO R1
      ;
 004F        AM2903 ,R10,R2,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910 CRTN,
ONE
      /    & MEM & MPR ; R10 TO R2, RETURN FROM SUBROUTINE
      ;
 0050 XCH23: AM2903 ,R2,R10,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R2 TO R10
      ;
 0051        AM2903 ,R3,R2,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R3 TO R2
      ;
 0052        AM2903 ,R10,R3,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910 CRTN,
ONE
      /    & MEM & MPR ; R10 TO R3, RETURN FROM SUBROUTINE
      ;
 0053 XCH34: AM2903 ,R3,R10,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R3 TO R10
      ;
 0054        AM2903 ,R4,R3,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910
      /    & MEM & MPR ; R4 TO R3
      ;
 0055        AM2903 ,R10,R4,AB,PASSR,YBUS,WRITEB & AM2901 & AM2910 CRTN,
ONE
      /    & MEM & MPR ; R10 TO R4, RETURN FROM SUBROUTINE
      ;
 0056 EXIT:  AM2903 & AM2910 & AM2901 & MEM & MPR
      ;
      END
0000 1110001000000000 00000XXXX0010011 1100XXXXXXXXXX1 00011001100X0000 00
0001 1110000000000000 00110XXXXXXXX011 0000XXXX10111000 00111001001X0000 00
0002 1110000000000000 1001000000001110 0100XXXXXXXXXX1 00011001100X0000 11
0003 0011001000000101 01100XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
0004 0001000000000010 00010XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
0005 0001000000000010 0001000010001100 0110000110100000 11011000100X0000 11
0006 1110XXXXXXXXXXX XXXX00010XXXX101 10001010XXXX0000 11011001100X0000 01
0007 1110XXXXXXXXXXX XXXX000100010100 0110000110100000 01011001100X0000 01
0008 1110XXXXXXXXXXX XXXX000100010100 01101010XXXX0000 11011001100X0000 01
0009 1110XXXXXXXXXXX XXXX000100010100 0110000110100000 01011001100X0000 01
000A 1110000000000000 00110XXXXXXX011 0000XXXX10111000 00111001001X0000 00
000B 0011110100000000 11100XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
000C 1110XXXXXXXXXXX XXXX000100010100 01101010XXXX0000 11011001100X0000 01
000D 0011000000000101 0110000100010100 0110000110100000 01011001100XXXXX 01
000E 1110000000000000 1101000000001110 0100XXXXXXXXXX1 00011001100X0000 11
000F 1100000000000000 01001XXXXXXX011 0000XXXX10111000 00111001000X0000 00
0010 0011000100000001 10100XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
0011 0001000000000010 10110XXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XX
0012 1110XXXXXXXXXXX XXXX000010001100 0110001010100000 11011000100X0000 11
0013 0001000000000010 10110XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
0014 1110XXXXXXXXXXX XXXX000100010100 01101010XXXX0000 11011001100X0000 01
0015 1110XXXXXXXXXXX XXXX000100010100 0110001010100000 01011001100X0000 01
0016 XXXX000000000000 0100X00010001110 0100XXXXXXXXXX1 00011001100X0000 11
0017 1110000000000000 0101000010001110 0100XXXXXXXXXX1 00011001100X0000 11
0018 1001XXXX00000001 00010XXXXXXX011 0000XXXXXXXXXX1 00011001100X0000 00
```

```
0019 1110000000000000 0101000010001110 0100XXXXXXXXXX1 0001100110OX0000 11
001A 0001000000000010 00010XXXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
001B 1110XXXXXXXXXXX XXXX0000010001100 0110000110100000 1101100010OX0000 11
001C 0001000000000010 00010XXXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
001D 1110XXXXXXXXXXX XXXX0000100010100 0110101OXXXX0000 1101100110OX0000 01
001E 1110XXXXXXXXXXX XXXX0000100010100 0110000110100000 0101100110OX0000 01
001F 1110XXXXXXXXXXX XXXX0000100010100 0110101OXXXX0000 1101100110OX0000 01
0020 0001000000000101 0110000100010100 0110000110100000 0101100110OX0000 01
0021 1110000000000000 0101000010XXX110 00000000XXXX0010 1100000110OX0000 11
0022 1110000000000000 1010000010XXX110 00000001XXXX0010 1101100010OX0000 11
0023 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000XXX000100010 1001100010OX0000 00
0024 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000600010000 0101100100IX0000 00
0025 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0026 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000100100000 0101100100IX0000 00
0027 0001110100000100 11010XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0028 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000000010000 0101100100IX0000 00
0029 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
002A 1010000OXXXXXXX XXXX0XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
002B 1110000000000000 0101000010XXX110 0000XXXX00000010 1001100010OX0000 11
002C 1110000000000000 1010000010XXX110 0000XXXX00010010 1001100010OX0000 11
002D 1110000000000000 1111000010XXX110 0000XXXX00100010 1001100010OX0000 11
002E 1110000000000001 0100000010XXX110 0000XXXX00110010 1001100010OX0000 11
002F 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000XXXX01000010 1001100010OX0000 00
0030 1110111111111111 11110XXXXXXX011 0000101001001001 1000001100OX0000 00
0031 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000000010000 0101100100IX0000 00
0032 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0033 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000100100000 0101100100IX0000 00
0034 0001110100000100 11010XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0035 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001000110000 0101100100IX0000 00
0036 0001110100000101 00000XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0037 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001101000000 0101100100IX0000 00
0038 0001110100000101 00110XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0039 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000XXXX10100000 1001100100OX0000 00
003A 1010001OXXXXXXX XXXX0XXXXXXX011 0000101010100000 0101100010OX0000 00
003B 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000000010000 0101100100IX0000 00
003C 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
003D 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000100100000 0101100100IX0000 00
003E 0001110100000100 11010XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
003F 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001000110000 0101100100IX0000 00
0040 0001110100000101 00000XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0041 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000XXXX10100000 1001100100OX0000 00
0042 1010001OXXXXXXX XXXX0XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0043 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000000010000 0101100100IX0000 00
0044 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0045 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000100100000 0101100100IX0000 00
0046 0001110100000100 11010XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0047 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000000010000 0101100100IX0000 00
0048 0001110100000100 10100XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
0049 1010000OXXXXXXX XXXX0XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
004A 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000010100000 1101100010OX0000 00
004B 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000100000000 1101100010OX0000 00
004C 1010000OXXXXXXX XXXX0XXXXXXX011 0000101000010000 1101100010OX0000 00
004D 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000000110100000 1101100010OX0000 00
004E 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001000010000 1101100010OX0000 00
004F 1010000OXXXXXXX XXXX0XXXXXXX011 0000101000100000 1101100010OX0000 00
0050 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001010100000 1101100010OX0000 00
0051 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001100100000 1101100010OX0000 00
0052 1010000OXXXXXXX XXXX0XXXXXXX011 0000101000110000 1101100010OX0000 00
0053 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000001110100000 1101100010OX0000 00
0054 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000010000110000 1101100010OX0000 00
0055 1010000OXXXXXXX XXXX0XXXXXXX011 0000101001000000 1101100010OX0000 00
0056 1110XXXXXXXXXXX XXXX0XXXXXXX011 0000XXXXXXXXXX1 0001100110OX0000 00
```